

GPO PRICE \$ _____

CFSTI PRICE(S) \$ _____

Hard copy (HC) 3.00

Microfiche (MF) 65

FINAL REPORT

NASA Contract NAS8-21059 ff 653 July 65

A Study of Fluid and Structural Behavior Under
Low Gravity Conditions

Submitted by Victor J. Law, Principal Investigator

for Tulane University, New Orleans, La.

ABSTRACT

A generalized method called Rotational Discrimination for nonlinear regression and equation solving is described. An adaptive method for system identification is then described for two types of situations; namely, time domain data or frequency domain data may be used. A method is also described for reducing certain types of time domain data into frequency form. Digital computer programs are listed and their usage is described for each of these procedures.

FACILITY FORM 502

N 68-24322
(ACCESSION NUMBER)

77
(PAGES)

CI-61780
(NASA CR OR TMX OR AD NUMBER)

(THRU)

1
(CODE)

19
(CATEGORY)



I. Rotational Discrimination for Nonlinear Regression and Equation Solving.

The method of Rotational Discrimination (RD) was first described by Fariss and Law (1). A brief description of the algorithm will now be given.

RD belongs to a class of methods such that the computations always begin from a point \underline{x}^0 in n - dimensional space and a move or increment, $\underline{\Delta x}$, is computed such that $\underline{x}^1 = \underline{x}^0 + F\underline{\Delta x}$ forms a search vector along which a "better" point is sought. This logic is repeated until no further improvement is possible. The choice of the scalar F is made by a one-dimensional search procedure.

The success of the method depends on a property of the $\underline{\Delta x}$ vector which shall be called truncation convergence. An algorithm for minimization has this property if, for sufficiently small F , the objective function $Q(\underline{x}^0 + F \underline{\Delta x})$ takes on a smaller value than $Q(\underline{x}^0)$. What this means is that $\underline{\Delta x}$ must point in a direction such that Q decreases at least locally. Hence, a better point can always be found by truncating F to some small positive value.

RD uses a unique combination of the method of scaled steepest descent and the Gauss-Newton method to minimize a sum of squares function.

A brief review of these methods will now be given in the sequel.

I. 1. Formulation as a Minimization Problem. The specific problem to which attention is now given is that of finding the value of an n -vector \underline{x} such that the functions

$$f_j(\underline{x}) = 0; \quad j = 1, 2, \dots, n \quad (1-1)$$

are satisfied. This problem may easily be formulated as a minimization problem by forming the sum of squares of residual as an objective function.

$$Q = \frac{1}{2} \sum_{j=1}^n f_j^2 \quad (1-2)$$

Clearly, if Q is minimized to its absolute minimum of zero then a solution to the original problem has been obtained.

The nonlinear regression problem may also be expressed as a sum of squares minimization problem with

$$f_j = g_j(\underline{x}) - \overset{\Delta}{g}_j; \quad j = 1, 2, \dots, n_d \quad (1-3)$$

where

n_d = number of data points

$g_j(\underline{x})$ = computed value of a function for the j th data point

$\overset{\Delta}{g}_j$ = experimental value of the function for the j th data point

\underline{x} = an n -vector of regression coefficients

I. 2. Unscaled Steepest Descent. Perhaps the oldest and still very popular method for unconstrained minimization is the method of steepest descent (SD). Strictly speaking, this method is a continuous one rather than a discrete one in that the path of steepest descent is a continuous curve. For practical use, however, the direction of steepest descent is found at the base point, \underline{x}^0 , and this direction is used to form the search vector.

The direction of steepest descent is given by

$$\underline{\Delta x} = - \frac{\partial Q(\underline{x}^0)}{\partial \underline{x}} \quad (1-4)$$

The search vector then becomes

$$\underline{x} = \underline{x}^0 + h \underline{\Delta x}$$

The value of h is usually selected by performing a one-dimensional search for a minimum in Q with respect to h. Perhaps the most popular one-dimensional search is the Golden Section Search (2). A more sophisticated method is described by Fariss and Law (3). It is also possible to simply find a value of h for which Q is smaller than at the base point rather than finding the minimum. There is no generally "best" procedure to use. In order to begin the search for h, however, it is necessary to select some value of this parameter as a first estimate. It is efficient to select this value of h based on the optimal h from the previous iteration. One means of accomplishing this is to use the following relationship:

$$\frac{h_o^{(i+1)}}{h_o^{(i)}} = \exp \left[0.88255 \tan^{-1} (0.56654 \ln F) \right] \quad (1-5)$$

where

$$F = \frac{h_{opt}^{(i)}}{h_o^{(i)}} \quad (1-6)$$

and $h_o^{(i+1)}$ = first estimate of optimal h for (i+1) th iteration

$h_o^{(i)}$ = first estimate of optimal h for i th iteration

$h_{opt}^{(i)}$ = optimal h for the i th iteration

This formula is purely empirical and merely attempts to update h_0 based on past experience while not allowing large changes in h_0 from one iteration to the next. The previous h_0 is multiplied by a factor between $\frac{1}{4}$ and 4, with $\frac{1}{4}$ corresponding to $F = 0$ and 4 to $F = \infty$. If $F = 1$, h_0 is not changed.

The Δx vector of Equation (1-4) is normal to the objective function contour at the base point and is guaranteed to have the truncation convergence property.

The most serious drawback of the method of steepest descent is the zig-zag tendency especially when near the solution. This property is best explained in the two-dimensional case. Referring to Figure 1-1, it is easily seen that successive directions of steepest descent will be orthogonal (perpendicular in 2-dimensions).

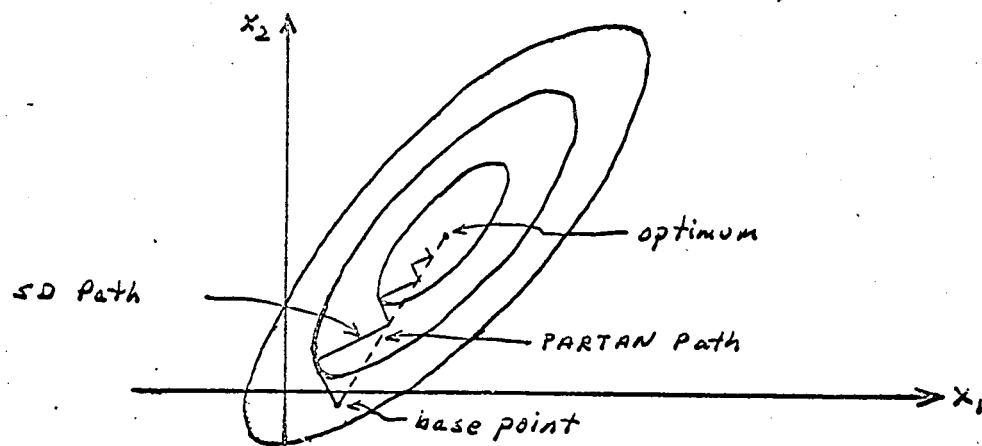


Fig. 1-1. Steepest descent for a quadratic function.

In order to overcome this difficulty, several techniques have been suggested [see, for example, Booth (3)]. One particular modification is the Method of Parallel Tangents developed by Shah, Buehler and Kempthorne (4).

Scaled steepest descent is another modification and will be described in the sequel.

I. 3. The Gauss-Newton Method (GN). This method uses Newton's method as a basis. That is, the equation for determining $\underline{\Delta x}$ is given by

$$\frac{\partial^2 Q}{\partial \underline{x}^2} \underline{\Delta x} = \frac{\partial Q}{\partial \underline{x}} \quad (1-7)$$

A useful and effective approximation is made for the Hessian, $\frac{\partial^2 Q}{\partial \underline{x}^2}$, however. In terms of the sum of squares Q function,

$$\frac{\partial Q}{\partial x_i} = \sum_{j=1}^n f_j \frac{\partial f_j}{\partial x_i} \quad (1-8)$$

and

$$\frac{\partial^2 Q}{\partial x_i \partial x_k} = \sum_{j=1}^n \left[\frac{\partial f_j}{\partial x_i} \frac{\partial f_j}{\partial x_k} + f_j \frac{\partial^2 f_j}{\partial x_i \partial x_k} \right] \quad (1-9)$$

Now, for equation solving the $f_j \rightarrow 0$ as the solution is approached. For regression problems, the f_j do not tend to zero but they tend to distribute about zero, some being positive and some being negative. Therefore, the so-called Gauss-Newton approximation is to omit the second term in the expression for $\frac{\partial^2 Q}{\partial x_i \partial x_k}$. That is,

$$\frac{\partial^2 Q}{\partial x_i \partial x_k} \approx \sum_{j=1}^n \frac{\partial f_j}{\partial x_i} \frac{\partial f_j}{\partial x_k} = G_{ik}$$

Thus, the relation for step size determination becomes

$$G \underline{\Delta x} = - \frac{\partial Q}{\partial \underline{x}} \quad (1-10)$$

In the absence of singularity of the G matrix, the (GN) procedure is very efficient at converging to the solution from a near point (i.e., where the sum of squares function becomes nearly quadratic and the f_j are small). This behavior could be anticipated from the approximate quadratic representation of Q by using the G matrix as an approximation to $\partial^2 Q / \partial \underline{x}^2$. Thus, quadratic convergence is obtained in the neighborhood of the solution.

As a protection against very long search trials, it is customary to limit the length of the search vector to some arbitrary value. While this will usually force convergence, it has one distinct disadvantage. In order to clearly understand why this is so, consider a problem where only one variable causes the singularity. In such a case the moves predicted for all other variables would be quite good. Thus, the truncation of the entire search vector is clearly inefficient in that only the move in the maverick variable need be truncated. This justifies the need for the following two operations:

- (1) Sorting those variables which cause the singularity in G.
- (2) Selectively truncating only the moves for these variables.

More will be said about this later.

The (GN) method is equivalent to the well known Newton-Raphson (5) (NR) method for equation solving. The (NR) method uses as its basis the linearization of each equation in the neighborhood of the base point. Thus,

$$f_j(\underline{x}) \approx f_j(\underline{x}^0) + \sum_{i=1}^n \frac{\partial f_j(\underline{x}^0)}{\partial x_i} \Delta x_i ; \quad j = 1, 2, \dots, n \quad (1-11)$$

The move is then determined so that each $f_j(\underline{x})$ would become zero if all functions were linear. In vector-matrix form, these relations become

$$J \underline{\Delta x} = -\underline{f} \quad (1-12)$$

where

J = the Jacobian matrix with $J_{ij} = \frac{\partial f}{\partial x_j}$

If equation (1-12) is premultiplied by J^T , there results

$$(J^T J) \underline{\Delta x} = - J^T \underline{f} \quad (1-13)$$

which is identical to Equation (1-11). That is,

$$G = J^T J \quad (1-14)$$

$$\frac{\partial Q}{\partial \underline{x}} = J^T \underline{f} \quad (1-15)$$

Equation (1-14) constitutes a proof that G is always at least positive semi-definite in that this property always holds for a product of a matrix and its transpose.

I. 4. Scaled Steepest Descent (SSD). It was also shown previously that the $\underline{\Delta x}$ vector given by Equation (1-14) does not, in general, pass through the minimum of a positive definite quadratic function with elliptical contours (see Fig. 1-1). A $\underline{\Delta x}$ vector which must pass through the solution point is given by Equation (1-7). If Q is a positive definite quadratic, then Equation (1-7) may be solved by diagonalizing the $\partial^2 Q / \partial \underline{x}^2$ matrix via eigenvalue - eigenvector calculations as follows:

$$S^T \frac{\partial^2 Q}{\partial \underline{x}^2} S S^T \underline{\Delta x} = - S^T \frac{\partial Q}{\partial \underline{x}} \quad (1-16)$$

Now let

$$S^T \frac{\partial^2 Q}{\partial \underline{x}^2} S = D \quad (\text{diagonal matrix with eigenvalues on the diagonals}) \quad (1-17)$$

$$\underline{y} = S^T \underline{\Delta x} \quad (1-18)$$

It is easily shown that

$$S^T \frac{\partial Q}{\partial \underline{x}} = \frac{\partial Q}{\partial \underline{y}} \quad (1-19)$$

Thus, Equation (1-16) becomes

$$D \underline{y} = \frac{\partial Q}{\partial \underline{y}} \quad (1-20)$$

Since D is diagonal its inverse is easily computed and

$$\underline{y} = D^{-1} \frac{\partial Q}{\partial \underline{y}} \quad (1-21)$$

It is desired now to find a further transformation of variables given by

$$\underline{z} = W \underline{y} \quad (1-22)$$

such that

$$\underline{z} = - \frac{\partial Q}{\partial \underline{z}} \quad (1-23)$$

(i.e., \underline{z} is calculated by unscaled SD)

The reason for desiring such a transformation is that \underline{z} is a vector which is a steepest descent vector passing through the solution.

To see what W must be, replace \underline{z} in Equation (1-23) by \underline{y} via the transformation, Equation (1-22). This leads to

$$W \underline{y} = - \frac{\partial Q}{\partial \underline{z}} = - W^{-1} \frac{\partial Q}{\partial \underline{y}} \quad (1-24)$$

The second equality is easily proven.

Solving Equation (1-24) for \underline{y} gives

$$\underline{y} = - W^{-2} \frac{\partial Q}{\partial \underline{y}} \quad (1-25)$$

Thus, in order that Equation (1-25) be identical with Equation (1-21), it is necessary that

$$W^{-2} = D^{-1} \quad (1-26)$$

or that

$$W = D^{\frac{1}{2}} \quad (1-27)$$

The last operation is legitimate since D and, consequently, W are both diagonal.

The conclusion here is that if each of the y_i are scaled by a factor of $\sqrt{d_{ii}}$ then the steepest descent vector in terms of these scaled variables will lead directly to the minimum of a positive definite quadratic function.

It is important to note that it is impossible, in general, to scale the Δx_i to make the SD search vector pass through the solution. To see this more clearly, consider a function with contours as shown in Fig. 1-2:

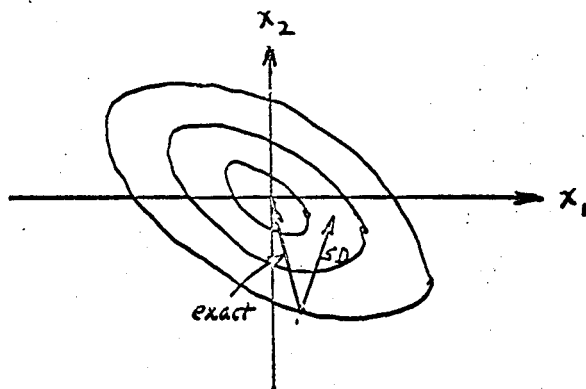


Fig. 1-2. SD and Exact Search Vectors.

The S-D direction produces positive values for both Δx_1 and Δx_2 whereas the exact search vector requires Δx_1 to be negative and Δx_2 to be positive. Thus, no positive scale factors exist. It is not proper to use negative scale factors since the property of truncation convergence would be destroyed. To see this, consider a diagonal matrix of scales, V . Then Equation (1-4) would be

$$\underline{\Delta x} = - V \frac{\partial Q}{\partial \underline{x}} \quad (1-28)$$

Then the directional derivative

$$\frac{dQ}{dh} = \left(\frac{\partial Q}{\partial \underline{x}} \right)^T \frac{d\underline{x}}{dh} = - \left(\frac{\partial Q}{\partial \underline{x}} \right)^T V \left(\frac{\partial Q}{\partial \underline{x}} \right) \quad (1-29)$$

Thus dQ/dh will be always negative only if V is positive definite. Hence, each scale factor must be positive.

Thus, scaling the \underline{y} vector is another means of accelerating the convergence of steepest descent. As long as the eigenvalues of the second derivative matrix are positive, the ideal scaling factors given by Equation (1-27) can be quite effective toward this end. However, if there is a large difference in the magnitudes of the eigenvalues or if any eigenvalues are negative or zero, then ideal scaling is either dangerous or not possible. In these situations it has been found that the concept of scaling the \underline{y} vector is still useful but must be suitably modified. One practical means of accomplishing this is given in what follows. First, consider the definition of a set of scale factors such that

$$y_i / s_i = - \frac{\partial Q}{\partial y_i} ; \quad i = 1, 2, \dots, n \quad (1-30)$$

From Equation (1-21) it is obvious that

$$s_i = 1/d_{ii} \quad (1-31)$$

Absolute scaling is not really necessary but only relative scaling is required (i.e., relative to the largest d_{ii}). It is therefore convenient to specify the scale factor associated with the largest eigenvalue as unity. Then all other scale factors will be greater than or equal to unity and the relationship for scale factors may conveniently be written as

$$\frac{s_i}{s_1} = \frac{d_{11}}{d_{ii}} \quad (1-32)$$

or, since s_1 is taken as unity

$$s_i = d_{11}/d_{ii} ; \quad i = 1, 2, \dots, n \quad (1-33)$$

However, if d_{ii} approaches zero or becomes negative then the ideal scale factor cannot be used. It has been found useful to define non-ideal scale factors by means of the following scale smoothing equation which is purely empirical:

$$\ln s_i = \ln P \left[1 - \exp \left(-\frac{1}{\ln P} \ln \frac{d_{11}}{d_{ii}} \right) \right] ; \quad d_{ii} > 0 \quad (1-34)$$

where P = maximum scale factor relative to unity

If $d_{ii} \leq 0$, s_i is set equal to P .

Note that Equation (1-34) reduces to Equation (1-32) when d_{11}/d_{ii} is near unity. As d_{ii} gets large, s_i approaches P asymptotically. In practice a value of P of about 100 has been found to be satisfactory. The modified, scaled steepest descent (MSSD) procedure may now be outlined completely as follows:

- (1) Select a distance factor h and a base point \underline{x}^0 .
- (2) Compute $\frac{\partial Q(\underline{x}^0)}{\partial \underline{x}}$ and $\frac{\partial^2 Q(\underline{x}^0)}{\partial x_2}$
- (3) Find the eigenvalues and eigenvectors of the second derivative matrix.
- (4) Compute a set of scale factors from Equation (1-34)
- (5) Apply (MSSD) to find the rotated variables, \underline{y} , as follows:
 - (a) Set $|y_1| = h$
 - (b) Compute for $1 < i \leq n$ all remaining y_i relative to y_1 as follows:

$$y_1 = -h \operatorname{sgn} \left(\frac{\partial Q}{\partial y_1} \right) \quad (1-35)$$

$$y_i = y_1 s_i^2 \frac{\partial Q}{\partial y_i} / \frac{\partial Q}{\partial y_1} ; \quad i = 2, 3, \dots, n \quad (1-36)$$

If any element of Equation (2.5-19) exceeds h in magnitude, it is truncated to have magnitude h .

- (6) Perform a one-dimensional search along the vector

$$\underline{x} = \underline{x}^0 + F \underline{\Delta x} \quad (1-37)$$

where

$$\underline{x} = S \underline{y}$$

F = a scalar parameter (note that this is the same F as in Equation (1-6)).

That is, vary F until Q is minimized.

- (7) Update h based on the experience of the one-dimensional search by applying Equation (1-5).
- (8) Go to step (2) and repeat until convergence is achieved, that is, until either of the following occur:
 - (a) the change in Q between iterations is within a tolerance
 - (b) the vector $F \underline{\Delta x}$ becomes very small
 - (c) each $\frac{\partial Q}{\partial x_i}$ becomes negligibly small.

I. 5. Rotational Discrimination (RD) . This method begins with the (GN) formula for step size determination given by Equation (1-10) and repeated below:

$$G \underline{\Delta x} = - \frac{\partial Q(\underline{x}^0)}{\partial \underline{x}} \quad (1-38)$$

First, the $\underline{\Delta x}$ coordinates are rotated as follows:

$$S^T G S S^T \underline{\Delta x} = - S^T \frac{\partial Q(\underline{x}^0)}{\partial \underline{x}} \quad (1-39)$$

Letting

$$D \triangleq S^T G S \quad (1-40)$$

and

$$\underline{y} \triangleq S^T \underline{\Delta x} \quad (1-41)$$

Equation (1-39) may be written as

$$D \underline{y} = - \frac{\partial Q}{\partial \underline{y}} \quad (1-42)$$

It is assumed that Equations (1-42) have been ordered such that d_{11} is the largest positive eigenvalue and each successive d_{ii} is algebraically smaller than the previous.

The (RD) method is basically a tactic for choosing a superior search vector in terms of the rotated coordinates, y , which exhibit zero local interaction. (RD) logic presupposes that reasonable external scaling of the variables has been accomplished so that the following assumptions are likely to be valid:

- (1) The acceptability of a move calculated by (NM) can be judged from its length -- long moves are suspect.
- (2) In the Gauss-Newton matrix, eigenvalues which are several orders of magnitude smaller than the largest are indicative of a linear dependence of Q on the associated y coordinate. Negative eigenvalues are associated with those y coordinates for which the objective function tends to exhibit a local maximum rather than a minimum.

The (RD) method then attempts to combine the best features of (GN) and (MSSD) by computing the y vector components discriminantly. That is, if the (GN) move for a particular y_i is not too long and if the associated eigenvalue is positive and not greatly different from d_{11} , then the (GN) calculation is used. Otherwise, (MSSD) is used. The complete logic for the (RD) minimization algorithm is as follows:

- (1) Select a base point, \underline{x}^0 , and a maximum allowable distance factor h .
- (2) Compute $\frac{\partial Q(\underline{x}^0)}{\partial \underline{x}}$ and G .
- (3) Find the eigenvalues and eigenvectors of G . Order D and S of Equation (1-40) so that the d_{ii} are in descending algebraic order.

- (4) Starting with y_1 , the elements of \underline{y} are computed by (GN) logic, that is

$$y_i = - \frac{\partial Q}{\partial y_i} / d_{ii} \quad (1-43)$$

until either

(a) $i = n$

(b) $d_{ii} \leq 0$

(c) $y_i > h$

- (5) When the (GN) sequence is terminated for reasons (b) or (c) above, at the k th parameter, then a switch is made to (MSSD) logic. For the k th parameter, y_k is then assigned a scale factor of 1 and

$$y_k = -h \operatorname{sgn}\left(\frac{\partial Q}{\partial y_k}\right) \quad (1-44)$$

- (6) If there are parameters in the list after the k th one, the (MSSD) sequence is continued until the end of the parameter list is reached. Thus for $i = k + 1, k + 2, \dots, n$,

$$y_i = y_k s_i \frac{\partial Q}{\partial y_i} / \frac{\partial Q}{\partial y_k} \quad (1-45)$$

The scale factors, s_i , are determined relative to $s_k = 1$ using Equation (1-34). If any y_i calculated by Equation (1-45) is larger than h in magnitude then it is truncated to have magnitude h .

If any $d_{ii} = 0$, then that y_i is set to zero. The logic behind this is discussed in the next section.

- (7) The resulting \underline{y} vector is converted back into a $\underline{\Delta x}$ vector by

$$\underline{\Delta x} = \underline{S} \underline{y} \quad (1-46)$$

and a one-dimensional search along the vector

$$\underline{x} = \underline{x}^0 + F \underline{\Delta x} \quad (1-47)$$

is performed,

- (8) The distance factor h , is updated using Equation (1-5) if any variables have been found by (MSSD). Otherwise, no updating is performed.
- (9) Convergence is achieved when either
 - (a) the change in Q between several successive iterations is within a tolerance.
 - (b) $F \Delta x$ becomes very small.
 - (c) each $\frac{\partial Q}{\partial x_i}$ become very small.
 - (d) if, for regression problems each y_i becomes statistically insignificant (see discussion below).

If convergence is not met, go to Step (2).

In regression problems, there is no absolute measure which constitutes a solution of the problem. This difficulty can be overcome by making use of the statistical nature of the problem. Of particular interest is the estimate of the residual variance given by

$$\sigma_f^2 = \frac{Q}{n_d - n} \quad (1-48)$$

It is then possible to define an estimate of the parameter variance as

$$\sigma_{y_i}^2 = \sigma_f^2 / d_{ii} \quad (1-49)$$

where

d_{ii} = the i th eigenvalue of the associated Gauss-Newton matrix, G .

With these statistical estimates defined, it is now possible to give modifications to the (RD) algorithm as just described for regression problems. First, if a parameter move cannot be justified on statistical grounds, then it is classified as a null effect parameter. That is, if the parameter variance given

by Equation (1-49) is larger than some arbitrary value (say 1.0) and if the parameter step (y_i) as computed by (GN) [i.e., Equation (1-43)] is less than the parameter variance then the parameter step is set to zero. Second, the regression process may be terminated based on the following test: for all transformed parameters which are not null effect, if the $y_i < 0.1\sigma_{y_i}$ then the algorithm is terminated.

I. 6. Singularity of G and Null Effect Variables. The singularity of the G matrix is usually caused by a phenomenon which shall be called null effect. Those variables which cause this condition will be called null effect variables. Null effect occurs when perturbation of a parameter or of some linear combination of parameters has no significant effect on any of the residuals in the sum of squares function. In a well posed problem, null effect should not, of course, be present at the solution point. For systems of nonlinear equations, however, null effect is common at points removed from the solution. This is caused by local redundancies or inconsistencies in the linearized versions of the equations. In either case, two or more linear equations become parallel to each other and hence no solution exists. As an example, consider the problem illustrated in Fig. 1-3.

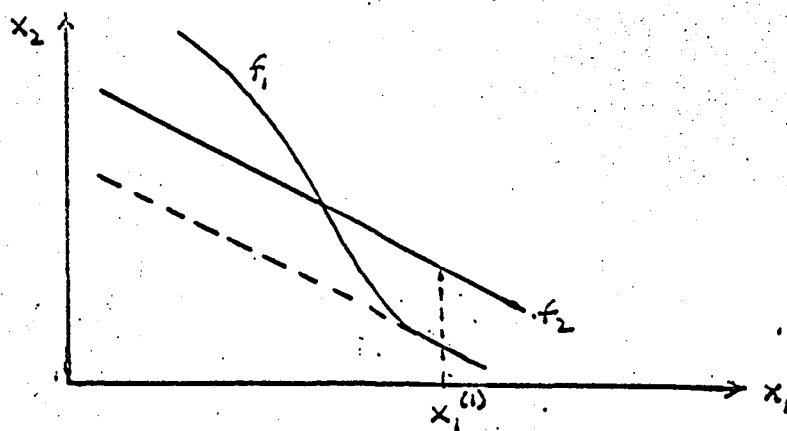


Figure 1-3. Illustration of null effect

I. 7. Rules for Using Subroutine RDRG. A FORTRAN IV subroutine named RDRG has been written which may be called to perform the regression or equation solving calculations using RD logic. The necessary rules for calling the subroutine from a main program or other subroutine are given below.

The basic characteristics of the system are as follows:

- 1) The user sets up arrays containing starting values and scaling factors for the parameters being sought, and then starts rotational discrimination action by the appropriate subroutine call.
- 2) Thereafter, until the iterative procedure is terminated by the subroutines, the values contained in the parameter array are under control of the subroutines. The user must provide programming to calculate residuals for the data points or equations for any set of values in the parameter array, on demand by the rotational discrimination algorithm.
- 3) When control is released, the parameter array will contain "best" values, i.e., those which cause the sum of squares of the errors to be a minimum.
- 4) In the form provided rotational discrimination operates as a restrained minimization algorithm -- limits are recognized for parameter values.
- 5) In its standard form, rotational discrimination obtains partial derivatives of the residuals automatically by numerical differentiation.
- 6) There are no arbitrary internal limits on the size of problem due to dimensioning, etc. The standard version requires approximately 9,000 words of 7044 core storage for program decks, plus the additional space necessary to hold internally generated arrays. As a general rule, problems involving up to about 70 parameters can be handled on a 32K 7044.

- 7) Nesting, e.g., using these programs as part of the calculation of residuals to be used by these programs, is permitted only if the user follows certain procedures to insure self-protection.
- 8) No special system features are required. Floating point numbers in intermediate output are produced by LPE format.
- 9) Large problems (e.g., more than 40 parameters) may cause the regression subroutines to make use of scratch storage on two utility units. Normally tapes 1 and 2 are used, however, alternates may be designated.
- 10) This system can be used in a chained job, provided certain minor limitations on overlay are observed.

I. 7.1 Programming for all-FORTRAN-IV Usage. A skeletal illustration of a program which meets the basic requirements for all-FORTRAN normal usage is shown below:

```

1  DIMENSION X(...), SCALE(...), Y(...), T(...), XMIN(...), XMAX(...), MDEX(...),
    D2QX(...)
2  DIMENSION S(...), SX(...), DER2(...), XX(...), MDE(...), DER1(...), DXBAR(...),
    DXX(...), DQ(...), DQX(...), DY(...), D2Q(...), SCALR(...)
3  DIMENSION XSTART(135)
4  COMMON /REGCOM/ COM(12), KCOM(22)
    EQUIVALENCE (KCOM(5), ICDE), (KCOM(7), IPT)
    .....
    X( ) =
    SCALE( ) =          (Put values in X, SCALE, XMIN and XMAX arrays)
    XMIN( ) =
    XMAX( ) =
    .....
5  CALL REGTAP (NUNIT1, NUNIT2)
6  CALL REGSET (NPT, NV, IPRINT, MAXCNT, TIMAX, KTAP, KBASE)
10 CALL RDRG (S, SX, DER2, D2QX, XX, MDE, DER1, DXBAR, DXX, DQ, DQX, DY, DY(KBASE),
    X, SCALE, XMIN, XMAX, D2Q, SCALR, MDEX, XSTART, DELTA)

```

```
11 IF (KCOM(5).EQ.0) GO TO 100
```

```
.....
```

```
DELTA =
```

```
GO TO 10
```

```
100 .....
```

```
(Regression or equation-solving finished)
```

Statements 1, 2 and 3 established dimensioning for arrays required by the subroutine system. Minimum space requirements for all of these, except DY, are outlined below. NVMAX is the maximum number of parameters expected in any use during the job run.

```
S, SCALE, MODE, DER1, DXBAR, DX, DQ, DQX, D2Q, SCALR, XMIN, XMAX, }NVMAX
MDEX,
```

```
S, SX DER2, D2QX
```

```
}(NVMAX)2
```

```
Y, T
```

```
}NPTMAX
```

XSTART must always have dimension 135.

The regression (or equation-solving) calculation is initiated by calling subroutine RDRG, statement 10. However, prior to this call five things must have been done:

- 1) Put starting values (initial guesses) for parameters in X array. Also, the lower bounds and upper bounds must be put into the XMIN and XMAX arrays, respectively.
- 2) Put scaling factors for parameters in SCALE array. (Suggestions concerning choice of scale factors for parameters are presented in Section I.7.2.)

Notes: NVMAX = maximum number of variables in any call

NPTMAX = maximum number of data points

Y = Array of dependent variable data points (there could be a Y₁, Y₂, etc.)

T = Array of independent variable data points

DELTA = (Calculated Y - data Y) for the IPTth data point and the present values in the X array.

- 3) Designate scratch utility units by calling REGTAP (statement 5). Arguments NUNIT1 and NUNIT2 are FORTRAN utility numbers. If scratch utility storage is not to be used, these arguments should be zero.
- 4) Set other required internal constants by calling REGSET (statement 6).
- 5) It is convenient to equivalence KCØM (5) to ICØDE and KCØM (7) to IPT.

The values for NPT, NV, IPRINT, MAXCNT, and TIMAX are constants selected by the user:

- | | |
|--------|---|
| NPT | = Number of data points to be used in a regression. In an equation-solving problem this is the number of equations to be solved. In general, it is the number of error terms whose sum of squares is to be minimized. |
| NV | = Number of unknown parameters to be sought in the regression of equation-solving problem, e.g., the number of elements in the X (and SCALE) array. In an equation-solving problem NV and NPT should be equal. |
| IPRINT | = Internal print control (see Section I.7.3) |
| MAXCNT | = Maximum allowable number of iterations based on fresh derivatives. |
| TIMAX | = Maximum allowable time (minutes) for regression. |

MAXCNT and TIMAX should be chosen* to prevent excessive "spinning." Argument

* It is extremely unlikely that more than 20 iterations will be required unless the problem is very poorly formulated. The "average" problem requires 5. Time requirements depend on the complexity of the residual calculation. However, as a rough guide, typical times have varied from 0.05 minutes for a 2 parameter problem to one hour for a 70 parameter problem. Note: The time limiter depends on subroutine TIMEX, which is provided in dummy form. An active TIMEX must be provided if the TIMAX parameter is to be of significance.

KTAP should be zero for standard usage. Argument KBASE is an index computed by REGSET for use in the call of RDRG.

The dimensioning required for array DY, depends on whether utility storage is used to reduce core storage needs:

	<u>No Utility Storage</u>	<u>Utility Storage Used</u>
Minimum DY dimension	(NPTMAX)*(NVMAX+4)	NVMAX+4

NPTMAX is the maximum number of data points (or equations) expected in any use during the job run.

The action required after return from subroutine RDRG depends on the value given to (ICØDE) by the subroutines:

When ICØDE = 0, the subroutines have completed their operation

and the calling program can proceed (statement 100). If a normal finish was obtained, KCØM(8) will be returned as zero. At this point the X array will contain the parameter values corresponding to the problem solution, and the final sum of residuals squared will be in CØM(9).

Non-normal finishes are indicated as described in section I.7.4.

When ICØDE > 0, the calling program must calculate the value of the residual for the IPTth point (or equation) and place this value in DELTA. The calculation must be made using the values of the parameters contained at that moment in the X array. Then the subroutines are returned to

Note: For equation solving an additional subroutine, REGCØN, must be called immediately following the call of REGTAP. The call should be made as follows:

CALL REGCØN(2.0,TØL, 0.0, 0.001, 0.05, 0.2, 10.0,2.0,1) where TØL = tolerance on sum of squares. That is, when CØM(9) \leq TØL, the calculations will be complete. A value for TØL of about 10^{-10} for well-scaled equations is recommended.

action by going back to the call of RDRG.

If it is desired to start another regression, constants should be reset by calling REGSET before the initial RDRG call.

I. 7.2 Selecting Scaling Factors for Parameters. Scale factors for the parameters must be placed in the SCALE array prior to the start of the RD process. These factors are used by the RD subroutines in calculating parameter increments for numerical differentiation, and as an assist to the RD logic in choosing the optimum search trajectory in the parameter space. It is strongly recommended that careful thought be given to their selection.

The initial numerical differentiation step uses as parameter increments the parameter scale factors multiplied by a constant, FDERIV, which is normally 0.001. Ideally, these increments should cause an average change in the residuals which is noticeable in the last 3 or 4 significant figures. Thus, as a first rule for obtaining scale factors:

- 1) Estimate the parameter increment size which will produce typically a residual change in the last 3 or 4 significant figures.
- 2) Divide this increment by FDERIV (0.001).

A second possible rule, which may be useful in some regression problems, is based upon the null effect parameter analysis which is built into RD logic. This logic prevents statistically insignificant changes in parameters when the standard deviation of estimation for the parameter exceeds a constant, FDSCRD, times the scale factor. FDSCRD is normally 1.0. This rule for scale factor selection may be constructed as follows:

- 1) Estimate the standard deviation of estimation for a parameter, assuming all other parameter values known, which would make modification of this parameter value by regression meaningless.
- 2) Divide this standard deviation by FDSCRD (1.0).

The above rules will usually suffice for choosing parameter scaling factors in problems for which the parameter values either have physical significance or are known in advance to lie within a certain range.

I. 7.3 Internal Print Control and Debugging. The call of REGTAP always produces a line of output which indicates the utilities used for temporary storage, if any:

UTILITIES XX AND XX USED FOR TEMPORARY STORAGE

The RD subroutines brought into action by the call of RDRG furnish considerable information about operation during use if the contained print statements are not suppressed. Suppression can be accomplished by setting IPRINT=0 in the call of RDRG. A limited amount of printing is obtained if IPRINT=1. However, when new problems are run, it is strongly recommended that the complete internal print facilities be used by setting IPRINT=2. The reason for this is that the output thus obtained is extremely useful in debugging the problem setup.

Typical examples of output are shown on the following pages. The sequence is as follows:

- 1) Heading line
- 2) Table of starting values & scaling factors for parameters
- 3) Table of controls (NVAR = number of parameters being solved for, NCASE = number of residuals)
- 4) Special option list
- 5) Iteration number list
- 6) Parameter, Internal variable table
- 7) Comments
- 8) SRCHS (one-dimensional search) table
- 9) Termination explanation

When IPRINT = 2, items 5 through 8 are repeated for each iteration. For IPRINT = 1, these are produced only for the last iteration.

The special option list contains the following two lines.

NO SHORTCUT ON JACOBIAN

BASE CASE RECALCULATION SKIPPED

The items in the Iter No line are as follows:

ITER NO M/N M is a count, starting with zero, of the number of iterations in which the determination of a search vector was based on a fresh evaluation of derivatives. N is set to zero when such an iteration is currently in progress. N is incremented by one for each iteration which does not base its search vector on fresh derivatives, in a sequence following an iteration which does evaluate derivatives. Thus 3/2 under ITER NO means the second iteration without derivative reevaluation following the fourth iteration which did involve derivative reevaluation. Iterations without derivative evaluation are used only in the accelerated mode, which is indicated in the special option list by the line:

SHORTCUT ON JACOBIAN

INACTIVE This number indicates how many "null effect parameters" have been identified and are not being used as variables in the current iteration. Additional information is furnished under INACTIVE pertaining to parameters at their limits.

DERIV This number indicates the nature of the differentiation process used by RD the last time derivatives were obtained:

2 - Numerical by central differences

1 - Numerical by forward differences

DIST This is the distance of the desired first try along the search vector, in a scaled parameter space. Thus, if the search increment vector is $\Delta X_1, \Delta X_2, \dots$, and parameter scale factors are s_1, s_2, \dots , DIST is equal to

$$\sqrt{\sum (\Delta X_i / s_i)^2}$$

FSD This is the current value of the steepest descent distance factor, used by RD logic as the maximum acceptable increment for any coordinate in the scaled, rotated space.

SIGMA This is the current value for the sum of residuals squared.

Under PARAMETERS are three columns giving respectively the parameter number, the current value of the parameter, and the parameter increment corresponding to the desired first try along the search vector.

Under INTERNAL VARIABLES are four columns, all referring to the coordinate system obtained after rotation (\bar{X}):

a) The Gauss-Newton matrix diagonal:

$$\sum_r (\partial y_r / \partial \bar{X}_i)^2 \quad (y_r = \text{rth residual})$$

b) The partial derivative of the sum of squares:

$$\partial Q / \partial \bar{X}_i$$

c) The coordinate increment corresponding to the desired first try along the search vector.

d) The mode of computation for the coordinates:

- 0 - Gauss-Newton
- 1 - Steepest Descent
- 2 - Truncated Steepest Descent
- 1, -2 - Not incremented (null effect)

The Iter No line and the Parameter, Internal variable table are printed in the middle of each iteration after a search vector has been determined, but

before the search has been started. If the search is conducted, the next output will be the SRCHS table, containing three columns:

- a) FX - The factor multiplied by the search vector increment to obtain a trial point. One is printed for each point tried.
- b) One of the following comments, indicating the subsequent action to be taken by the one-dimensional search subroutine (SRCHS):

TRY OPTIM
NG, STOP
CUT STOPPED
CUT
OPTIM OK
TAKE BEST
TAKE PREV
TRY MORE
QUIT ON PREV
QUIT ON LOW
REVERSE
NO GO
NO, TRY MORE
TAKE

- c) The value for the sum of residuals squared corresponding to the FX value.
(The initial value in the column is for the base point, FX = 0.)

The final value in the FX column is the one accepted to determine a new base point.

Instead of proceeding with the search, the iteration may be repeated, with a preliminary line of explanation:

CONVERTED TO CENTRAL DIFFERENCES (Numerical differentiation is to

be repeated using a central difference rather than forward difference method - this will not cause a change under ITER No.)

The RD program may terminate operation of the problem at this point, with one of the following lines of explanation:

R.D. FINISH NORMAL

R.D. HAS REACHED MAX NUMBER OF ITERATIONS

R.D. TERMINATED DUE TO LAST STEP BEING ABNORMALLY SMALL

Termination may also occur at the end of the search, after printing of the SRCHS table, with the line:

R.D. TIME LIMIT EXCEEDED

Further explanations of termination are given in Section I.7.4.

The nature of errors in problem setup can frequently be determined by an analysis of the internal output furnished by the RD programs. Following is a suggested scheme for analysis.

- 1) Examine the tables of starting values, scales, and control parameters to see that this information has been transmitted correctly to the RD subroutines. Note particularly that NVAR = number of parameters to be solved for, and NCASE = number of residuals.
- 2) Check SIGMA. A zero value for SIGMA in the first iteration line usually means the residual calculation is fouled up.
- 3) Check the first column under Internal Variables. These are the eigenvalues of the Gauss-Newton matrix. If all are zero, the derivative calculation is not responding to parameter changes. Very small numbers here, with corresponding -1 or -2 in the right hand column, indicate null effect parameters, which may reflect difficulties in the problem setup.

- 4) Check the ACTION column in the SRCHS table. Any of the following occurring on an iteration where M/N under Iter NO has N = 0 is indicative of trouble:

NØ, TRY MØRE

NØ GØ

REVERSE

CUT STØPPED

Possible sources of trouble are "noise" in the residual evaluation, or overwriting of computer values due to inadequate dimensioning in the using program. "Noise" usually manifests itself only as the solution is approached.

If an equation-solving application fails to yield a suitable low value for the sum of residuals squared, the last iteration should show at least one very small number in the first column under INTERNAL VARIABLES. This indicates that a non-zero minimum has been reached -- either a solution to the equations does not exist, or it cannot be reached from the starting point used.

I. 7.4 Types of Finishes. Completion of solving action is indicated by return of ICØDE, as 0. The reason for termination is indicated by the integer value placed in the element KCØM(8) in named common REGCØM according to the following table:

<u>KCØM(8)</u>	<u>TERMINATION REASON</u>
0	Normal finish - all convergence requirements satisfied.
1	Maximum number of iterations using fresh derivatives has been completed. This maximum number is specified to the solving subroutines by the 4th argument in the call of REGSET, MAXCNT.

- 2 Time limit has been exceeded. This limit is specified to the solving subroutines by the 5th argument in the call of REGSET, TIMAX.
- 3 Frustration finish -- all convergence requirements not satisfied, but progress has slowed to less than tolerable limits.

A KCØM(8) signal of 0 (normal finish) indicates that the answer should be completely satisfactory in all respects. A KCØM(8) signal of 3 (frustration finish) indicates that no further meaningful reduction in sum of squares is possible, but the parameter values either do not satisfy all requirements or could be significantly refined further by additional (probably excessive) computation time.

When KCØM(8) signals of 1 or 2 are encountered, a best solution has not as yet been reached, due to arbitrary interruption by time or iteration count limits. Such interruption could be prevented or delayed by using larger limit values for MAXCNT or TIMAX.

Frustration finishes are usually caused by one of three situations:

- 1) An equation-solving application in which solution is impossible, or prevented by hanging up at a minimum in the sum of squares other than zero.
- 2) "noise" in the functional evaluation.
- 3) Regression involving several parameters not clearly defined by the data.

The first of these situations always produces at least one null-effect variable, recognizable as a MØDE of -1 or -2. There is no point in forcing further RD action from the same starting point.

Situation (2) is usually recognizable as indicated in diagnostic item 5 in Section I.7.3. The best remedy is to repair the functional evaluation.

Situation (3) is usually recognizable from the presence of several variables still in steepest descent (MØDE 1 or 2) at exit.

A frustration finish will also usually occur if one attempts regression on data which give an exact fit.

I. 7.5 Calculating All Residuals For The Same Parameter Vector at The Same Time. In some instances it may be desirable to calculate all residuals for the same parameter vector before proceeding to the next parameter vector. Since the normal request sequence involves covering all parameter vectors before moving on to the next residual, this tactic is best accomplished by doing all the residual evaluations while the regression subroutines are requesting only the first (IPT=1), and storing for recall. The following skeletal program illustrates this procedure, with the residual values stored in DARRAY:

```
DIMENSION DARRAY(...,...)
10 CALL RDRG....
11 IF (ICØDE.EQ.0) GØ TØ 100
    IF (IPT.GT.1) GØ TØ 30
    DØ 20 I = 1,NPT
    .....
20 DARRAY(I,ICØDE) =
30 DELTA = DARRAY(IPT,ICØDE)
    GØ TØ 10
100 .....
```

As before, in an all-Fortran system ICODE, INEW, and IPT would be replaced by KCOM(5), (6), and (7). The minimum dimensions[†] for DARRAY are NPT x (2*NV+1).

References For Section I

1. Fariss, R.H. and V.J. Law, "Practical Tactics for Overcoming Difficulties in Non-linear Regression and Equation Solving," to appear in SIAM Journal on Applied Math (1967).
2. Wilde, D.J., "Optimum Seeking Methods," Prentice-Hall (1964).
3. Booth, A.D., "Numerical Methods," Butterworths (1957).
4. Shah, B.V., R.J. Buehler and O. Kemthorne, J. SIAM, 12, 74 (1964).
5. Rosenbrock, H.H. and C.Storey, "Computational Techniques for Chemical Engineers," p. 59, Pergamon Press (1966).

† If the derivative taking is restricted to forward differences, the minimum dimensions are NPT x (NV+1).

II. Adaptive Model Development Using Time Domain Data

II.1. Summary

A technique has been developed for model updating or identification cast in the time domain. This presented technique uses dynamic time domain input-output data to find a linear constant coefficient second order system which best represents the data in a least squares sense. It incorporates rotational discrimination for the nonlinear regression analyses.

II.2. Mathematical Development

II.2.1 The Simulated Model

For a general linear second order system disturbed by an excitation E for a time duration t , the following ordinary differentiation equation (O.D.E.) may be written:

$$\frac{\tau^2 d^2 y}{dt^2} + 2\xi\tau \frac{dy}{dt} + y = E(t) \quad (1)$$

where:

y = Dependent output magnitude

t = Independent time

τ = System time constant

ξ = system damping ratio

In the formulated technique, the excitation E of the system was obtained as a function of time by approximating $E(t)$ with straight line segments of the form

$$E(t) = c_i(t-t_i) + d_i \quad (2)$$

where

t_i = time at the beginning of the i th interval

c_i = slope of the excitation in the time interval

$$t_i \leq t \leq t_i + \Delta t_i$$

d_i = intercept of the excitation in the same time interval

This approach is called the piecewise linear approximation (PLA) method. The Δt_i intervals are not necessarily equal and should be chosen so that approximate representation is reasonable.

The eigenvalues of the characteristic equation of Equation (1) may be of three types depending upon ξ . For each type, respective solutions may be calculated as follows for $t_i \leq t \leq t_i + \Delta t_i$

- 1) Solution of (1) with complex eigenvalues ($\xi < 1$.)

$$y(t) = e^{(SS)t} [(P-B)\cos(RR)t + c_1 \sin(RR)t] + At + B \quad (3)$$

and

$$\begin{aligned} \frac{dy(t)}{dt} &= (SS)e^{(SS)t} [(P-B)\cos(RR)t + c_1 \sin(RR)t] + A + \\ &\quad e^{(SS)t} [-RR(P-B)\sin(RR)t + (RR)c_1 \cos(RR)t] \end{aligned} \quad (4)$$

where:

SS = Real part of eigenvalue

$$= -\frac{\xi}{\tau}$$

RR = Complex element of eigenvalue

$$= \frac{(\sqrt{1-\xi^2})}{\tau}$$

$$c_1 = \frac{Q-A-SS(P-B)}{RR}$$

Q = Derivative of response at t_i

P = Response y at t_i

A = Slope of excitation approximated by P.L.A. = c_i

B = Intercept of excitation by P.L.A.
 $= 2\tau\xi A$

2) Solution of (1) with multiple eigenvalues ($\xi=1.0$).

$$y(t) = (P-B)e^{D_1 t} + [D_1(B-P)-A+Q]t e^{D_1 t} + At + B \quad (5)$$

and

$$\frac{dy(t)}{dt} = D_1(P-B)e^{D_1 t} + D_1[D_1(B-P)-A+Q]t e^{D_1 t} + [D_1(B-P)-A+Q]e^{D_1 t} + A \quad (6)$$

where

$$D_1 = -\frac{1}{\tau}$$

3) Solution of (1) with real and distinct eigenvalues ($\xi>1.0$).

$$y(t) = \left\{ P-B + \frac{[Q-A+D_1(B-P)]}{D_1-D_2} \right\} e^{D_1 t} + \left[\frac{Q-A+D_1(B-P)}{D_2-D_1} \right] e^{D_2 t} + At + B \quad (7)$$

and

$$\frac{dy(t)}{dt} = D_1 \left\{ P-B + \frac{[Q-A+D_1(B-P)]}{D_1-D_2} \right\} e^{D_1 t} + D_2 \left[\frac{Q-A+D_1(B-P)}{D_2-D_1} \right] e^{D_2 t} + A \quad (8)$$

where

$$D_1 = -\frac{\xi}{\tau} + \frac{\sqrt{\xi^2-1}}{\tau}$$

$$D_2 = -\frac{\xi}{\tau} - \frac{\sqrt{\xi^2-1}}{\tau}$$

These Equations (2-8) were incorporated into the simulated model of a linear constant coefficient second order system with parameters τ and ξ . With these parameters given, and the excitation E given by P.L.A., a complete response profile was generated and used to back identify parameters, namely τ and ξ .

II.2.2 The Adaptive Model Generator Program for Linear Constant Coefficient Second Order Systems.

This package identifies the parameters of the candidate system or model. Input-output response data taken from the actual system are used as input data for this program. The program then, through nonlinear regression analyses, determines parameters τ and ξ based on the minimization of the sum of the squares of the differences in the output of the actual system and of a simulated second order model. This difference is defined as a "residual."

- 1) Use of program 2141 M and subroutines 2141A and 2141B.
 - a) Function of and definitions of parameters in program 2141M not including those set for R-D.

Program 2141M is the main program of this algorithm. This subroutine reads in the excitation magnitudes, excitation times, number of excitation data points, response magnitudes, response times, number of response data points, initial guesses for parameters, and number of parameters. This subroutine also calls rotational discrimination for regression and related subroutines. Definitions of variables are

N = number of coefficients
M = number of response data points
NM = number of excitation data points

X0 = initial guesses for coefficients (τ and ξ)

Y = magnitude of response

T = output time

E = magnitude of excitation

TT = input time

b) Function of subroutine 2141A (SECORD).

This subroutine furnishes to the main program residuals whose sum is to be minimized. These residuals are named DARRAY (IPT, I CODE) in this subroutine and are the only output from this subroutine.

c) Function and output explanation of subroutine 2141B (PRINTD).

Following the identification of the candidate model, the regression will terminate. Regressed values for τ and ξ will be given and eigenvalue type will be stated. A complete response profile of the candidate model will be generated and printed at the same response time intervals as were read into the program. Residuals will then be provided along with the sum of the squares of the residuals. Tables I-A, I-B, and I-C give example information for all possible eigenvalue types

TABLE I-A

	<u>Tau</u>	<u>Eta</u>
Actual Parameter Values	1.0	.1
Starting Parameter Values	1.4	.21
Calculated Parameter Values	.999994	.100007

Number of iterations = 5

Sum of squares of residuals = $.593 \times 10^{-10}$

TABLE I-B

	<u>Tau</u>	<u>Eta</u>
Actual Parameter Values	1.0	1.0
Starting Parameter Values	1.2	1.2
Calculated Parameter Values	1.000001	.999999

Number of iterations = 4

Sum of squares of residuals = $.147 \times 10^{-11}$

TABLE I-C

	<u>Tau</u>	<u>Eta</u>
Actual Parameter Values	1.0	2.0
Starting Parameter Values	1.10	2.20
Calculated Parameter Values	1.000005	1.999991

Number of iterations = 3

Sum of squares of residuals = $.184 \times 10^{-11}$

III. Adaptive Model Development Using Frequency Domain Data

III.1. Summary

This section presents an algorithm developed for model identification in the frequency domain. The algorithm makes use of general Bode diagrams taken from a simulated linear constant coefficient second order system. Two options are presented in that phase angle data may or may not be utilized as supplemental information in updating or identifying candidate models. It has been found that phase angle data can be of significant value in more accurately updating or identifying candidate models. Rotational discrimination is used in the nonlinear regression analyses.

III.2 Mathematical Development

The algorithm for adaptive model development utilizing complete Bode diagram information for a linear constant coefficient second order system may be formulated using the unique relationship between time domain error and frequency domain error as shown by Schnelle (2). That is

$$\Psi = \sum_{\substack{\text{all data} \\ \text{sets}}} \sum_{i=1}^N \sum_{j=1}^M (Y_{ij} - Z_{ij})^2 \quad (1)$$

where

M = the number of discrete time points

N = the number of output variables of the system

Y_{ij} = the value of y_i at the j th discrete time level

Z_{ij} = the value of z_i at the j th discrete time level

is equivalent to

$$\begin{aligned} \Psi' = & \sum_{\text{all data sets}} \sum_{i=1}^N \sum_{j=1}^P [(G_{ik} \cos \theta_{ik} - H_{ik} \cos \phi_{ik})^2 \\ & + (G_{ik} \sin \theta_{ik} - H_{ik} \sin \phi_{ik})^2] \end{aligned} \quad (2)$$

Note: Data set in the time domain is defined as Z_i for one test or experiment corresponding to a particular value of time. Data set in the frequency domain is defined as Z_i for one test or experiment corresponding to a particular value of frequency.

where

P = the number of discrete frequencies considered

G_{ik} = calculated magnitude ratio for the i th variable at the k th frequency level

H_{ik} = data magnitude ratio for the i th variable at the k th frequency level

θ_{ik} = calculated phase angle for the i th variable at the k th frequency level

ϕ_{ik} = data phase angle for the i th variable at the k th frequency level

III.2.1 Formulation of adaptive model development using only magnitude ratio data from the bode diagram (programs 2111M, 2111A, and 2111B).

The magnitude ratio and phase angle for a linear constant coefficient second order system are given by

$$\text{Magnitude ratio} = \frac{1}{\sqrt{(1-\omega^2\tau^2)^2 + (2\xi\tau\omega)^2}} \quad (3)$$

$$\text{and phase angle} = \tan^{-1} \left(\frac{-2\xi\tau\omega}{1-(\omega\tau)^2} \right) \quad (4)$$

These may be obtained from the transfer function

$$G(s) = \frac{1}{\tau^2 s^2 + 2\xi\tau s + 1} \quad (5)$$

In equations (3) and (4), τ and ξ are parameters with ω being the frequency. Utilizing equation (3) only and rotational discrimination as the regression algorithm, a complete program for updating or identifying candidate models was formulated. Various Bode diagrams of second order systems have been investigated, and candidate models have been generated by the proposed procedure

in test cases. Among these test cases was a model with complex distributed parameters.

It should also be pointed out that a gain factor has been incorporated into this algorithm as an additional unknown parameter.

- 1) Function of and definitions of parameters in program 2111M not including those set for R-D.

Program 211M is the main program of this algorithm. This subroutine reads in the magnitude ratios, frequencies, number of data points, initial guesses for parameters, and number of parameters. This main program also calls rotational discrimination for regression and related subroutines.

Definitions of variables are

N = actual number of coefficients

M = actual number of data points

X0 = initial guesses for τ , ξ and the gain respectively

Y = magnitude ratio

Z = frequency.

- 2) Definitions of variables and functions of subroutine 2111A.

This subroutine furnishes the residuals to program 2111M used in the regression algorithm. This is the only output of subroutine 2111A.

- 3) Definitions of variables and function of subroutine 2111B.

This subroutine is a post-regression program which furnishes information concerning the regressed function and parameters. Among this information are the regressed values of τ , ξ and the gain of the candidate system. Next, magnitude ratios are calculated and printed. A magnitude is printed at each input frequency. Other information includes the magnitude ratio differences

between the candidate system and the simulated linear constant coefficient second order system. Note further that in subroutine 2111B, a normalized sum of squares is calculated. This sum of squares may give insight into the regression for gains much less or much greater than one.

III.2.2 Formulation of adaptive Model Development Using All Available Data From the Bode Diagram (Programs 2121M, 2121A, and 2121B).

The major difference in model development using all bode diagram data and only the magnitude ratio is that there are two (2) sum of squares terms a the function to be minimized. This is seen in Equation (2) and leads to no particular problem. The RD regression programs can easily handle this situation. The magnitude ratio and phase angle for the simulated system are given in Equations (3) and (4). With these equations and Equation (2) the required residual may be obtained.

In this case, with the phase angle data being considered, the regression produced better parameter values for τ , ξ , and the gain in test cases.

1) Function of and definitions of parameters in program 2121M not including those set for R-D.

Program 2121M is the main program in this algorithm. This program reads in the magnitude ratios, phase angles, number of data points, frequencies, initial guesses for parameters, and the number of parameters. This subroutine also calls rotational Discrimination for regression and related subroutines. Definitions of variables are

N = actual number of parameters

M = actual number of data points

XO = initial guesses for coefficients or parameters

Y = magnitude ratio

ANG = phase angle in degrees

Z = frequency

2) Definition of variables and function of subroutine 2121A.

The function of this program is identical to that of subroutine 2111A as explained in Section III.2.1.

3) Definitions of variables and function of subroutine 2121B.

The function of this subroutine is comparable to subroutine 2111B. Subroutine 2121B adds additional information in that phase angles are calculated and printed as are the magnitude ratios.

III.2.3 Extension of this technique to a linear constant coefficient with order system.

The transfer function for this general system may be written as

$$G(s) = \frac{1}{a_N s^N + a_{N-1} s^{N-1} + \dots + a_1 s + a_0} \quad (6)$$

This gives a magnitude ratio of

$$\text{Mag. Ratio} = \left| \frac{1}{a_N (j\omega)^N + a_{N-1} (j\omega)^{N-1} + \dots + a_1 j\omega + a_0} \right| \quad (7)$$

and a phase angle of

$$P.A. = \left[\frac{1}{a_N(j\omega) + a_{N-1}(j\omega)^{N-1} + \dots + a_1 j\omega + a_0} \right] \quad (8)$$

Here parameters include the a_i 's. The basic theory behind the formulation of a set of residuals whose sum is to be minimized depends upon Equations (2), (7), and (8). No programs are provided for this. However, they could easily be generated if a model of higher order is believed necessary.

The following tables give example information produced by the programs described above.

TABLE II

	<u>MAGNITUDE RATIO DATA ONLY</u>		
	<u>Tau</u>	<u>Eta</u>	<u>Gain</u>
Actual parameter values	1.0	.1	1.0
Starting parameter values	.98	.4	.89
Calculated parameter values	.99989	.09976	.99814
Number of iterations = 7			
Sum of squares of residuals = $.46 \times 10^{-4}$			

TABLE III

	<u>ALL BODE DIAGRAM INFORMATION</u>		
	<u>Tau</u>	<u>Eta</u>	<u>Gain</u>
Actual parameter values	1.00	.10	1.0
Starting parameter values	.98	.40	.89
Calculated parameter values	.999952	.099985	.999362
Number of iterations = 7			
Sum of squares of residuals = $.983 \times 10^{-4}$			

IV. Reduction of Pulse Test Data to Frequency Response Form.

IV.1 Introduction

The utilization of pulse testing procedures to experimentally determine system frequency response has become established as an efficient technique. The purpose of the investigation reported here was to compare two commonly used procedures for reducing experimental pulse test data to Bode diagram information. The two methods were compared for the case of a second order system, both with and without superimposed noise.

IV.2. Mathematical Development

For a process having the system transfer function $G(s)$, the response to an arbitrary input $F_i(s)$ is

$$F_o(s) = G(s) F_i(s) \quad (1)$$

where $F_i(s)$ is the Laplace transform of the input, and F_o is the Laplace transform of the system response. Thus, applying the definition of the Laplace transform, and using $s=j\omega$, equation (1) leads to

$$G(j\omega) = \frac{\int_0^{\infty} f_o(t) e^{-j\omega t} dt}{\int_0^{\infty} f_i(t) e^{-j\omega t} dt} \quad (2)$$

Pulse testing procedure involves using a finite pulse of limited duration to be applied as the input $f_i(t)$ to the system. From the time histories of input pulse $f_i(t)$ and the response or output pulse $f_o(t)$, equation (2) can be used to

determine the system dynamics.*

In this paper a "pulse function" is defined as a function, having arbitrary shape, that is zero at time zero, and assumes definite values for a finite length of time referred to as the pulse width T, after which it returns to zero and remains there indefinitely. From this definition it follows that Equation (2) can be rewritten as

$$G(j\omega) = \frac{\int_0^T f_o(t) e^{-j\omega t} dt}{\int_0^T f_i(t) e^{-j\omega t} dt} \quad (3)$$

An important advantage of pulse testing over other experimental procedures is that the data obtained from a single well-designed pulse test can be used in Equation (3) to obtain the transfer function G as a function of the frequency ω . Conversion of the complex number $G(j\omega)$ to polar form then yields the information from which a Bode diagram can be developed. Applying the identity

$$e^{-j\omega t} = \cos(\omega t) - j \sin(\omega t)$$

to both integrals in Equation (3) and reducing the results algebraically gives the magnitude and the phase angle of G as

$$|G| = \left[\frac{A_o^2 + B_o^2}{A_i^2 + B_i^2} \right]^{1/2} \quad (4)$$

$$\angle G = \tan^{-1} \left[\frac{A_o B_i - A_i B_o}{A_o A_i + B_o B_i} \right] \quad (5)$$

* Both $f_i(t)$ and $f_o(t)$ are actually assumed to be deviations from the steady state.

where

$$A = \int_0^T f(t) \cos(\omega t) dt \quad (6)$$

$$B = \int_0^T f(t) \sin(\omega t) dt \quad (7)$$

The computational problem, then, is seen to be that of evaluating the integrals A and B, given $f(t)$, for both input and output functions. Since $f_i(t)$ and $f_o(t)$ are obtained experimentally, their values are often known only at some set of discrete points between time zero and the respective pulse widths. Thus, a numerical method must be used to evaluate the integrals A and B.

Various numerical methods have been applied to this data reduction problem. Some of these, including the two used in this investigation, are discussed in papers by Clements and Schnelle (1) and Lees and Dougherty (2). In this investigation, two commonly used methods were compared under various circumstances.

The first of these is the application of the trapezoidal rule to the product functions that constitute the integrands in (6) and (7). In this case, the integrals are represented as

$$A = \frac{1}{2} \sum_{k=1}^{N-1} [f^{(k)}(t) \cos(\omega t_k) + f^{(k+1)}(t) \cos(\omega t_{k+1})] [t_{k+1} - t_k] \quad (8)$$

$$B = \frac{1}{2} \sum_{k=1}^{N-1} [f^{(k)}(t) \sin(\omega t_k) + f^{(k+1)}(t) \sin(\omega t_{k+1})] [t_{k+1} - t_k] \quad (9)$$

where N is the number of discrete points at which $f(t)$ is known, and $f^{(k)}(t)$ is the k th discrete value of $f(t)$. It is assumed that

$$f_1(t) = f_N(t) = 0$$

$$t_1 = 0, t_N = T$$

The formulas (8) and (9) do not require constant time interval sizes.

A major shortcoming of the trapezoidal rule is that the product curves $f(t) \sin(\omega t)$ and $f(t) \cos(\omega t)$ oscillate with increasing frequency as ω is increased. Thus, the approximation of these product curves will eventually deteriorate at some frequency, above which the results will be of no value.

The second method considered in this study will be referred to as piecewise linear approximation (PLA). It consists simply of approximation of the original pulse curves with straight-line segments, followed by analytical integration of the products of the approximations times $\sin(\omega t)$ and $\cos(\omega t)$. The working formulas for A and B in this case are

$$A = \frac{1}{\omega} \sum_{k=1}^{N-1} [f^{(k+1)} \sin(\omega t_{k+1}) - f^{(k)} \sin(\omega t_k)] + \frac{1}{\omega^2} \sum_{k=1}^{N-1} S_k [\cos(\omega t_{k+1}) - \cos(\omega t_k)] \quad (10)$$

$$B = -\frac{1}{\omega} \sum_{k=1}^{N-1} [f^{(k+1)} \cos(\omega t_{k+1}) - f^{(k)} \cos(\omega t_k)] + \frac{1}{\omega^2} \sum_{k=1}^{N-1} S_k [\sin(\omega t_{k+1}) - \sin(\omega t_k)] \quad (11)$$

$$S_k = \frac{f^{(k+1)} - f^{(k)}}{t_{k+1} - t_k} \quad (12)$$

The comments directly following Equations (8) and (9) apply to these formulas as well. An obvious advantage of the PLA method is that the accuracy of the results will be independent of the value of w .

The procedure for comparing the two methods described above required generating discrete values of the functions $f_i(t)$ and $f_o(t)$ without noise, and with two levels of superimposed Gaussian noise, using a digital computer. Two types of input pulse were used: a ramp function and a displaced cosine, defined respectively as

$$f_i(t) = \begin{cases} t/T_i & 0 \leq t \leq T_i \\ 0 & t \geq T_i \end{cases} \quad (13)$$

$$f_i(t) = \begin{cases} 1 - \cos \frac{2\pi t}{T_i} & 0 \leq t \leq T_i \\ 0 & t \geq T_i \end{cases} \quad (14)$$

The general second order transfer function

$$G(s) = \frac{1}{\tau^2 s^2 + 2\xi\tau s + 1} \quad (15)$$

General expressions for $f_o(t)$ were derived for each case. The criteria for the selection of the pulse width T_i and the number of discrete points N at which the functions $f_i(t)$ were evaluated are developed in section IV.5.

The procedure, then, consisted of using generated values of $f_i(t)$ and $f_o(t)$ to estimate the magnitude $|G|$ and the phase angle $\angle G$, as a function of frequency, by both trapezoidal integration and PLA. The two methods were then compared by observing the relative accuracy with which the true magnitudes and phase angles of (15) were reproduced over a range of frequencies.

IV.3 Results

The results of this study are now presented. Several conclusions can be drawn from the magnitude results. The most obvious observation is that the results obtained with trapezoidal integration begin to deviate quite badly from the true values at a frequency of about 2. Since the pulse widths of the input pulses, for both the ramp and the displaced cosine pulses, are about 2.5, this frequency corresponds to a sinusoid with a period slightly larger than the pulse widths. The maximum frequency considered corresponds to a sinusoid with a period slightly shorter than one-half of the pulse-widths. This deterioration of the results of trapezoidal integration is obviously due to the oscillation of the product curves $f(t) \sin(wt)$ and $f(t) \cos(wt)$ that was alluded to earlier.

A second observation is that, except for a few points, the results obtained with PLA are significantly more accurate than those obtained with trapezoidal integration. In addition, the PLA results do not deteriorate significantly with increasing frequency over the frequency range considered.

It was rather surprising to find that, with both 2 and 10 per cent superimposed noise, there are several points at which the PLA method gave more accurate values than were obtained with no noise. The explanation for this is that the PLA method has an inherent bias that is, to some extent, "wiped out" by introduction of random derivations from the true curve. The results at both noise levels show that, in general, noise does not have an overwhelming effect on results obtained with the PLA method. The final observation is that, in both cases without noise, trapezoidal integration seems to give significantly better results than the PLA method for the first five points. No explanation for this anomaly can be arrived at that would

be consistent with the absence of this effect in the presence of noise.

Conclusions similar to those drawn for magnitudes can be arrived at for phase angles.

IV.4. Program to Process Pulse Test Data by Trapezoidal Integration and Piecewise Linear Approximation

The integration formulas given in the text as equations (8) and (9) for trapezoidal integration, and equation (10), (11) and (12) for PLA, were programmed in a single FORTRAN program. The FORTRAN listing is given in **Appendix III**.

IV.5. Selection of Input Pulse Width from A Priori Knowledge of G(s)

It is clear that the numerical calculation of magnitudes and phase angles, using Equations (4), (5), (6) and (7) of the text, break down at these frequencies for which the modified Fourier integrals A_i and B_i (for the input function) are zero. In this work, it was arbitrarily decided that frequencies above the value at which the first zero occurs would not be considered. Theoretically, a ramp function produces no zeroes. However, the displaced cosine will give zeroes for

$$\omega + \frac{2n\pi}{T_i}, \quad n = 2, 3, \dots$$

Thus, a relationship between T_i and the largest frequency of interest, ω_{\max} , is

$$w_{\max} = \frac{4\pi}{T_i} \quad (16)$$

Note that the period of a sinusoid having this frequency will be exactly equal to one-half the pulse width T_i of the input pulse.

In addition, it was arbitrarily decided that the largest frequency that one might be interested in plotting on a Bode diagram should satisfy the relationship:

$$W_{\max} = 5\tau \quad (17)$$

where τ is the time constant (for a general second order system, say).

Now, combining (16) and (17), there results

$$T_i = \frac{.8\pi}{\tau} \quad (18)$$

To estimate a lower bound on the number of input function sample points, the following relationship proposed by Lees and Dougherty [see Reference (2)] is adopted

$$\Delta t \leq \frac{.2\pi}{w_{\max}} \quad (19)$$

Thus, if N is the number of sample points,

$$\frac{T_i}{N} \leq \frac{.2\pi}{w_{\max}} \quad (20)$$

which leads to

$$N < \frac{w_{\max}^T i}{2\pi} \quad (21)$$

Substituting (17) and (18) into (21), there results

$$N > \frac{(5\tau)(.8\pi)}{.2\pi}$$

or

$$N > 20$$

References

- (1) Clements, W.C. and K.B. Schnelle, IEC PD and D, 2, 94 (1963).
- (2) Lees, S. and R.C. Dougherty, J. Basic Engr., 445 (1967).

APPENDIX I
FORTRAN PROGRAM LISTINGS
FOR SECTION II

SIBFTC 2141M NODECK

DIMENSION X(6), ESX(6), XMIN(6), XMAX(6), XO(6), Y(140), T(140),
1 E(140), TT(140), DARRAY(140,13)

DIMENSION S(6,6), SX(6,6), DER2(6,6), XX(6), MODE(6), DER1(6),
1 DXBAR(6),DXX(6),DQ(6),DQX(6),DY(700),D2Q(6),SCALR(6),
2 MODEX(6),D2QX(6,6),XSIARI(135)

C
C
C
C
C
C
C
C
C
C
C
C
C

ESX = EXTERNAL SCALE FACTORS

Y = CALCULATED FUNCTION

T = INDEPENDENT VARIABLE

N = ACTUAL NO. OF COEFF'S

M= NO. OF RESPONSE POINTS

NM= NO. OF EXITATION POINTS

X = COEFF'S OF FUNCTION (TO BE CALCULATED)

XO = INITIAL GUESSES OF COEFF'S

999 READ 101, N, M, NM
WRITE (6,130)
130 FORMAT (/14H1 N M NM//)
WRITE (6,180) N,M,NM
180 FORMAT (1X,I2,3X,I2,3X,I2)
READ (5,102) (XO(I), I=1,N)
WRITE (6,120)
120 FORMAT (/16H INITIAL POINTS//)
WRITE (6,150) (XO(I), I=1,N)
READ (5,102) (ESX(I), I=1,N)
WRITE (6,121)
121 FORMAT (/24H EXTERNAL SCALE FACTORS//)
WRITE (6,150) (ESX(I), I=1,N)
READ (5,102) (XMIN(I), I=1,N)
WRITE (6,122)
122 FORMAT (/9H XMIN(S)//)
WRITE (6,150) (XMIN(I), I=1,N)
READ (5,102) (XMAX(I), I=1,N)
WRITE (6,123)
123 FORMAT (/9H XMAX(S)//)
WRITE (6,150) (XMAX(I), I=1,N)
READ (5,102) (Y(I) , I=1,M)
WRITE (6,124)
124 FORMAT (/6H Y(S)//)
WRITE (6,150) (Y(I), I=1,M)
READ (5,102) (T(I), I=1,M)
WRITE (6,125)
125 FORMAT (/6H T(S)//)
WRITE (6,150) (T(I), I=1,M)
READ 102, (E(I), I=1,NM)
PRINT 126
126 FORMAT (/6H E(S)//)
PRINT 150, (E(I), I=1,NM)
READ 102, (TT(I), I=1,NM)
PRINT 127
127 FORMAT (/7H TT(S)//)
PRINT 150, (TT(I), I=1,NM)
101 FORMAT (3I5)
102 FORMAT (10F8.4)
150 FORMAT (1H0,10F13.6)

```

129 DO 21 I=1,N
21 X(I)= XO(I)
C
EQUIVALENCE (KCOM(5), ICODE) , (KCOM(7), IPT)
C
COMMON/REGCOM/COM(12),KCOM(22)
C
NPT=M
NV=N
C
CALL REGTAP (0,0)
CALL REGSET (NPT, NV, 2, 50, 5.0, 0, KBASE)
C
959 CALL RDRG (S,SX,DER2,D2QX,XX,MODE,DER1,DXBAR,DXX,DQ,DQX,DY,
1      DY(KBASE),X,ESX,XMIN,XMAX,D2Q,SCALR,MODEX,XSTART,DELTA)
IF (ICODE) 100,100,20
20 IF ( IPT .GT. 1 ) GOTO 30
CALL SECORD (E, TT, NM, X(2), X(1), Y, T, M, ICODE, DARRAY)
30 DELTA= DARRAY (IPT, ICODE)
GOTO 959
100 CONTINUE
CALL PRINTD (X, N, Y, T, M, E, TT, NM)
GOTO 999
END

```

SIBFTC 2141A NODECK
 SUBROUTINE SECDORD (E, TT, NM, ETA, TAU, Y, T, NPT, ICODE,
 1 DARRAY)

C
 C THIS SUBROUTINE CALCULATES DARRAY(.....) FOR A SECOND
 C ORDER SYSTEM.

C
 C PARAMETERS OF THIS SECOND ORDER SYSTEM ARE ETA AND
 C TAU (DAMPING RATIO AND TIME CONSTANT).
 C

DIMENSION E(1), TT(1), Y(1), T(1), DARRAY(140,13)

C
 C
 C NOW TEST THE VALUE OF ETA.
 C

N= NM-1

C
 111 IF (ETA-1.) 21, 22, 23
 21 I=1
 IPT=1
 P=0.0
 Q=0.0
 SS= -ETA/TAU
 RR= ((1.-ETA**2)**.5)/TAU
 Z= T(1)
 501 IF (Z-TT(I+1)) 50, 50, 52
 50 TNT12 = Z - TT(I)
 KSET = 0
 53 S1= EXP(SS*TNT12)
 C= (E(I+1) - E(I))/(TT(I+1) - TT(I))
 B= E(I)-2.*TAU*ETA*C
 S2= P-B
 S3=(Q-C-SS*(P-B))/RR
 S4= SIN(RR*TNT12)
 S5= COS(RR*TNT12)
 Y1= S1*(S2*S5 + S3*S4) + C*TNT12 + B
 YY= S1*SS*(S2*S5 + S3*S4) + S1*(-RR*S2*S4 + RR*S3*S5) + C
 IF (KSET .GT. 0) GOTO 54
 DARRAY(IPT,ICODE)= Y(IPT)- Y1
 IPT=IPT+1
 IF (IPT-NPT) 51, 51, 56
 51 Z=T(IPT)
 GOTO 501
 52 KSET= 1
 TNT12= TT(I+1)-TT(I)
 GOTO 53
 54 P= Y1
 Q= YY
 I=I+1
 IF (I-N) 501, 501, 55
 55 Z= T(IPT)-TT(NM)

C
 C
 C AT THIS POINT, C(I) AND B(I) ARE BOTH 0.

H1= EXP(SS*Z)
 H3= (Q-SS*P) /RR
 H4= SIN(RR*Z)
 H2=P

```

H5= COS(RR*Z)
Y1= H1*(H2*H5 + H3*H4)
YY= SS*H1*(H2*H5 + H3*H4) + H1*(-RR*H2*H4 + RR*H3*H5)
DARRAY(IPT,ICODE)= Y(IPT)-Y1
IPT=IPT+1
IF (IPT-NPT) 55, 55, 56
56 GOTO 989
22 I=1
IPT=1
P=0.0
Q=0.0
D1= -1./TAU
Z=T(IPT)
601 IF (Z-TT(I+1)) 60, 60, 62
60 TNT12= Z - TT(I)
KSET =0
63 C= (E(I+1)- E(I))/(TT(I+1)- TT(I))
B= E(I)-2.*TAU*ETA*C
FF= P-B
C2= Q-C+D1*(B-P)
DDD=D1*TNT12 +1.
DED= EXP(TNT12*D1)
Y1= FF*DED+ C2*TNT12*DED+ C*TNT12 + B
YY=D1*FF*DED + C2*DDD*DED + C
IF (KSET .GT. 0) GOTO 64
DARRAY(IPT,ICODE)= Y(IPT)-Y1
IPT=IPT+1
IF (IPT-NPT) 61, 61, 66
61 Z= T(IPT)
GOTO 601
62 KSET=1
TNT12= TT(I+1)- TT(I)
GOTO 63
64 P=Y1
Q= YY
I=I+1
IF (I-N) 601, 601, 65
65 Z= T(IPT) - TT(NM)
C2= Q-D1*P
DSD= EXP(Z*D1)
Y1=P*DSD + C2*Z*DSD
YY= D1*P*DSD + D1*(-P)*D1*Z*DSD
DARRAY(IPT,ICODE)= Y(IPT)-Y1
IPT= IPT+1
IF (IPT-NPT) 65, 65, 66
66 GOTO 989
23 I=1
IPT=1
P=0.0
Q=0.0
D1= -ETA/TAU -((ETA**2-1.)**.5)/TAU
D2= -ETA/TAU +((ETA**2-1.)**.5)/TAU
Z=T(IPT)
701 IF (Z-TT(I+1)) 70, 70, 72
70 TNT12= Z - TT(I)
KSET= 0
73 C= (E(I+1)- E(I))/(TT(I+1)- TT(I))
B= E(I)-2.*TAU*ETA*C

```

```

      C2= (Q-C+D1*(B-P))/(D2-D1)
      C1= P- B+ (Q-C+D1*(B-P))/(D1-D2)
      DFD1=EXP(D1*TNT12)
      DFD2=EXP(D2*TNT12)
      Y1=C1*DFD1+ C2*DFD2+      C*TNT12 + B
      YY= D1*C1*DFD1+ D2*C2*DFD2+ C
      IF (KSET .GT. 0) GOTO 74
      DARRAY(IPT,ICODE)= Y(IPT)-Y1
      IPT=IPT+1
      IF (IPT-NPT) 71, 71, 76
71  Z=T(IPT)
      GOTO 701
72  KSET=1
      TNT12= TT(I+1)- TT(I)
      GOTO 73
74  P=Y1
      Q=YY
      I=I+1
      IF (I-N) 701, 701, 75
75  Z= T(IPT)-TT(NM)
      C1= P+(Q-D1*P)/(D1-D2)
      C2= (Q-P*D1)/(D2-D1)
      Y1=C1*EXP(D1*Z) + C2*EXP(D2*Z)
      YY=D1*C1*EXP(D1*Z) + D2*C2*EXP(D2*Z)
      DARRAY(IPT,ICODE)= Y(IPT)-Y1
      IPT=IPT+1
      IF (IPT-NPT) 75, 75, 76
76  GOTO 989
C
989  RETURN
C
      END

```

```

$IBFTC 2141B  NODECK
      SUBROUTINE PRINTD (X, N, Y, T, M, E, TT, NM)
C      THIS SUBROUTINE FURNISHES ADDITIONAL INFORMATION
C      CONCERNING THE REGRESSED FUNCTION AND PARAMETERS
      DIMENSION X(1), Y(1), T(1), E(1), TT(1), Y2(140), PP(140)
      WRITE (6,190)
190  FORMAT (//20H1REGRESSION COMPLETE//)
      WRITE (6,191)
191  FORMAT (//12H          TAU, 10X, 3HETA//)
      WRITE (6,150) (X(I), I=1,N)
150  FORMAT (1X, 10F13.6)
3200 FORMAT (1H0, F7.3, 8X, F14.7, 8X, F14.7)
162  FORMAT(//32H0Y1 CALCED AT EACH T FOR NEW X'S//)
      N= NM-1
      ETA=X(2)
      TAU=X(1)
      IF ( ETA - 1. ) 9, 99, 919
9  PRINT 3000
3000 FORMAT (1H0, 25H EIGNVALUES ARE COMPLEX //)
      WRITE (6,162)
      PRINT 3100
3100 FORMAT(4H0 ,1HT,12X,14H RESPONSE (Y1),6X,18H DERIVATIVE OF Y /)
      I=1
      IPT=1
      Z= T(IPT)
      P=0.0
      Q=0.0
      SS= -ETA/TAU
      RR= ((1.-ETA**2)**.5)/TAU
501  IF (Z-TT(I+1)) 50, 50, 52
50  TNT12 = Z - TT(I)
      KSET=0
53  S1= EXP(SS*TNT12)
      C= (E(I+1) - E(I))/(TT(I+1) - TT(I))
      B= E(I)-2.*TAU*ETA*C
      S2= P-B
      S5= COS(RR*TNT12)
      S3=(Q-C-SS*(P-B))/RR
      S4= SIN(RR*TNT12)
      Y1= S1*(S2*S5 + S3*S4) + C*TNT12 + B
      YY= S1*SS*(S2*S5 + S3*S4) + S1*(-RR*S2*S4 + RR*S3*S5) + C
      IF (KSET .GT. 0) GOTO 54
      PRINT 3200, Z, Y1, YY
      Y2(IPT)= Y1
      IPT=IPT+1
      IF (IPT-M) 51, 51, 56
51  Z=T(IPT)
      GOTO 501
52  KSET=1
      TNT12= TT(I+1)-TT(I)
      GOTO 53
54  P=Y1
      Q=YY
      I=I+1
      IF (I-N) 501, 501, 55
55  ZE= T(IPT)
57  Z=T(IPT)- TT(NM)

```

C
C

AT THIS POINT, C(I) AND B(I) ARE BOTH 0.

```
H1= EXP(SS*Z)
H3= (Q-SS*P) /RR
H4= SIN(RR*Z)
H2= P
H5= COS(RR*Z)
Y1= H1*(H2*H5 + H3*H4)
YY= SS*H1*(H2*H5 + H3*H4) + H1*(-RR*H2*H4 + RR*H3*H5)
PRINT 3200, ZE, Y1, YY
Y2(IPT)= Y1
IPT=IPT+1
ZE=T(IPT)
IF (IPT-M) 57, 57, 56
56 GOTO 96
99 PRINT 4000
4000 FORMAT (2H0 ,26H EIGNVALUES ARE MULTIPLE /)
WRITE (6,162)
PRINT 3100
I=1
IPT=1
Z=T(IPT)
P=0.0
Q=0.0
D1= -1./TAU
601 IF (Z-TT(I+1)) 60, 60, 62
60 TNT12= Z - TT(I)
KSET=0
63 C= (E(I+1)- E(I))/(TT(I+1)- TT(I))
B= E(I)-2.*TAU*ETA*C
FF= P-B
C2= Q-C+D1*(B-P)
DDD=D1*TNT12 +1.
DED= EXP(TNT12*D1)
Y1=FF*DED + C2*TNT12*DED+ C*TNT12 + B
YY=D1*FF*DED + C2*DDD*DED + C
IF (KSET .GT. 0) GOTO 64
PRINT 3200, Z, Y1, YY
Y2(IPT)= Y1
IPT=IPT+1
IF (IPT-M) 61, 61, 66
61 Z= T(IPT)
GOTO 601
62 KSET=1
TNT12= TT(I+1)-TT(I)
GOTO 63
64 P=Y1
Q= YY
I=I+1
IF (I-N) 601, 601, 65
65 ZE= T(IPT)
67 Z= T(IPT)- TT(NM)
C2= Q-D1*P
DSD= EXP(Z*D1)
Y1=P*DSD + C2*Z*DSD
YY= D1*P*DSD + D1*(-P)*D1*Z*DSD
PRINT 3200, ZE, Y1, YY
Y2(IPT)= Y1
```

```

      IPT=IPT+1
      ZE=T(IPT)
      IF (IPT-M) 67, 67, 66
66   GOTO 96
919  PRINT 5000
5000 FORMAT (2H0 ,35H EIGNVALUES ARE REAL AND DISTINCT /)
      WRITE (6,162)
      PRINT 3100
      I=1
      IPT=1
      Z=T(IPT)
      P=0.0
      Q=0.0
      D1= -ETA/TAU -((ETA**2-1.0)**.5)/TAU
      D2= -ETA/TAU +((ETA**2-1.0)**.5)/TAU
701  IF(Z-TT(I+1)) 70, 70, 72
70   TNT12= Z - TT(I)
      KSET=0
73   C= (E(I+1)- E(I))/(TT(I+1)- TT(I))
      B= E(I)-2.*TAU*ETA*C
      C2= (Q-C+D1*(B-P))/(D2-D1)
      C1= P- B+ (Q-C+D1*(B-P))/(D1-D2)
      DFD1=EXP(D1*TNT12)
      DFD2=EXP(D2*TNT12)
      Y1=C1*DFD1+ C2*DFD2+ C*TNT12 + B
      YY= D1*C1*DFD1+ D2*C2*DFD2+ C
      IF (KSET .GT. 0) GOTO 74
      PRINT 3200, Z, Y1, YY
      Y2(IPT)= Y1
      IPT=IPT+1
      IF (IPT-M) 71, 71, 76
71   Z=T(IPT)
      GOTO 701
72   KSET=1
      TNT12= TT(I+1)- TT(I)
      GOTO 73
74   P=Y1
      Q=YY
      I=I+1
      IF (I-N) 701, 701, 75
75   ZE= T(IPT)
77   Z= T(IPT)- TT(NM)
      C1= P+(Q-D1*P)/(D1-D2)
      C2= (Q-P*D1)/(D2-D1)
      Y1=C1*EXP(D1*Z) + C2*EXP(D2*Z)
      YY=D1*C1*EXP(D1*Z) + D2*C2*EXP(D2*Z)
      PRINT 3200, ZE, Y1, YY
      Y2(IPT)= Y1
      IPT= IPT+1
      ZE=T(IPT)
      IF (IPT-M) 77, 77, 76
76   GOTO 96
96   WRITE (6,163)
163  FORMAT (/23HODIFFERENCE IN Y1 AND Y//)
      DO 11 I=1,M
11   PP(I)= Y2(I) - Y(I)
      WRITE (6,150) (PP(I), I=1,M)
      SQRQ = 0.0

```

```
DO 12 I=1,M
12 SQRQ = SQRQ +(PP(I))**2
WRITE (6,152)
152 FORMAT (//28H0SUM OF SQUARES OF RESIDUALS//)
WRITE (6,151) SQRQ
151 FORMAT (1X, E15.6)
C
RETURN
C
END
```

APPENDIX II
FORTRAN PROGRAM LISTINGS
FOR SECTION III

SIBFTC 2111M NODECK

```
DIMENSION X(6), ESX(6), XMIN(6), XMAX(6), XO(6), Y(70), Z(70),
1Q(70), P(70)
DIMENSION S(6,6), SX(6,6), DER2(6,6), XX(6), MODE(6), DER1(6),
1 DXBAR(6), DX(6), DQ(6), DQX(6), DY(1680), D2Q(6), SCALR(6),
2 MODEX(6), D2QX(6,6), XSTAR(135)
```

C
C
C
C
C
C
C
C
C
C
C
C

```
ESX = EXTERNAL SCALE FACTORS
Y = CALCULATED FUNCTION
Z = INDEPENDENT VARIABLE
N = ACTUAL NO. OF COEFF'S
M = ACTUAL NO. OF DATA POINTS
X = COEFF'S OF FUNCTION ( TO BE CALCULATED )
XO = INITIAL GUESSES OF COEFF'S
```

```
999 READ (5,101) N,M
WRITE (6,130)
130 FORMAT (//8H1 N M//)
WRITE (6,180) N,M
180 FORMAT (1X,I2,3X,I2)
READ (5,102) ( XO(I), I=1,N )
WRITE (6,120)
120 FORMAT (//16H INITIAL POINTS//)
WRITE (6,150) (XO(I), I=1,N)
READ (5,102) ( ESX(I), I=1,N )
WRITE (6,121)
121 FORMAT (//24H EXTERNAL SCALE FACTORS//)
WRITE (6,150) (ESX(I), I=1,N)
READ (5,102) (XMIN(I), I=1,N )
WRITE (6,122)
122 FORMAT (//9H XMIN(S)//)
WRITE (6,150) (XMIN(I), I=1,N)
READ (5,102) (XMAX(I), I=1,N )
WRITE (6,123)
123 FORMAT (//9H XMAX(S)//)
WRITE (6,150) (XMAX(I), I=1,N)
READ (5,102) ( Y(I) , I=1,M )
WRITE (6,124)
124 FORMAT (//6H Y(S)//)
WRITE (6,150) (Y(I), I=1,M)
READ (5,102) (Z(I), I=1,M )
WRITE (6,125)
125 FORMAT (//6H Z(S)//)
WRITE (6,150) (Z(I), I=1,M)
101 FORMAT (2I4)
102 FORMAT (10F8.4)
150 FORMAT (1X, 10F13.8)
```

C
C
C
C
C
C
C

```
DO 2 I=1,N
2 X(I) = XO(I)

EQUIVALENCE (KCOM(5), ICODE) , (KCOM(7), IPT)

COMMON/REGCOM/COM(12),KCOM(22)

NPT=M
```

NV=N

C

CALL REGTAP (0,0)

CALL REGSET (NPT, NV, 2, 50, 5.0, 0, KBASE)

C

10 CALL RDRG (S,SX,DER2,D2QX,XX,MODE,DER1,DXBAR,DXX,DQ,DQX,DY,

1 DY(KBASE),X,ESX,XMIN,XMAX,D2Q,SCALR,MODEX,XSTART,DELTA)

C

IF (ICODE) 100,100,20

20 CALL F2MREG (Y, Z, X, DELTA, IPT)

GO TO 10

100 CONTINUE

CALL F2MCAL (X, Y, Z, Q, N, M, P)

GO TO 999

END

SIBFTC 2111A NODECK

 SUBROUTINE F2MREG (Y, Z, X, DELTA, IPT)

C THIS SUBROUTINE FURNISHES THE TRANSFER FUNCTION

C TO BE REGRESSED ON BY R - D

 DIMENSION X(1), Y(1), Z(1)

 DELTA1 = ((1.-(Z(IPT)*X(1))**2)**2 + (2.*X(2)*Z(IPT))**2)**.5

 DELTA = Y(IPT) - X(3)/DELTA1

C

 RETURN

C

 END

\$IBFTC 2121M NODECK

DIMENSION X(6), ESX(6), XMIN(6), XMAX(6), XO(6), Y(140), Z(140),
1Q(140), P(70), ANG(140), ANL(140)

DIMENSION S(6,6), SX(6,6), DER2(6,6), XX(6), MODE(6), DER1(6),
1 DXBAR(6),DXX(6), DQ(6),DQX(6),DY(3360),D2Q(6),SCALR(6),
2 MODEX(6), D2QX(6,6), XSTART(135)

C
C
C
C
C
C
C
C
C
C
C
C
C
C

ESX = EXTERNAL SCALE FACTORS

NMAX = MAX. NO. OF COEFF'S (6)

MMAX = MAX. NO. OF DATA POINTS (70)

Y = CALCULATED FUNCTION

ANG = REAL PHASE ANGLE IN DEGREES (A. R.)

Z = INDEPENDENT VARIABLE

N = ACTUAL NO. OF COEFF'S

M = ACTUAL NO. OF DATA POINTS

X = COEFF'S OF FUNCTION (TO BE CALCULATED)

XO = INITIAL GUESSES OF COEFF'S

999 READ (5,101) N,M
WRITE (6,130)
130 FORMAT (/8H1 N M//)
WRITE (6,180) N,M
180 FORMAT (1X,I2,3X,I2)
READ (5,102) (XO(I), I=1,N)
WRITE (6,120)
120 FORMAT (/16H INITIAL POINTS//)
WRITE (6,150) (XO(I), I=1,N)
READ (5,102) (ESX(I), I=1,N)
WRITE (6,121)
121 FORMAT (/24H EXTERNAL SCALE FACTORS//)
WRITE (6,150) (ESX(I), I=1,N)
READ (5,102) (XMIN(I), I=1,N)
WRITE (6,122)
122 FORMAT (/9H XMIN(S)//)
WRITE (6,150) (XMIN(I), I=1,N)
READ (5,102) (XMAX(I), I=1,N)
WRITE (6,123)
123 FORMAT (/9H XMAX(S)//)
WRITE (6,150) (XMAX(I), I=1,N)
READ (5,102) (Y(I) , I=1,M)
WRITE (6,124)
124 FORMAT (/6H Y(S)//)
WRITE (6,150) (Y(I), I=1,M)
READ (5,102) (ANG(I), I=1,M)
WRITE (6,126)
126 FORMAT (/19H ANG(S) IN DEGREES//)
WRITE (6,150) (ANG(I), I=1,M)
DO 23 I=1,M
23 ANG(I) = (ANG(I))/57.29578
READ (5,102)(Z(I), I=1,M)
WRITE (6,125)
125 FORMAT (/6H Z(S)//)
WRITE (6,150) (Z(I), I=1,M)
101 FORMAT (2I4)
102 FORMAT (10F8.4)
150 FORMAT (1X, 10F13.8)

C

```

      DO 21 I=1,N
21  X(I) = XO(I)
      DO 22 I=1,M
      MM = I + M
      Y(MM) = Y(I)
      ANG(MM) = ANG(I)
22  Z(MM) = Z(I)
      II = M/2
      IM = (3*M)/2

C
      EQUIVALENCE (KCOM(5), ICODE) , (KCOM(7), IPT)
C
      COMMON/REGCOM/COM(12),KCOM(22)
C
      NPT=M*2
      NV=N
      CALL REGTAP (0,0)
      CALL REGSET (NPT, NV, 2, 50, 5.0, 0, KBASE)
C
9  CALL RDRG (S,SX,DER2,D2QX,XX,MODE,DER1,DXBAR,DXX,DQ,DQX,DY,
1  DY(KBASE),X,ESX,XMIN,XMAX,D2Q,SCALR,MODEX,XSTART,DELTA)
C
      IF (ICODE) 100,100,20
20  CALL F2PREG ( Y, Z, X, DELTA, IPT,II, IM, M, ANG )
      GO TO 9
100 CONTINUE
      CALL F2PCAL( X, Q, ANL, Y, Z, N, M, P, II )
      GO TO 999
      END

```

\$IBFTC 2123A NODECK

SUBROUTINE F2PREG(Y, Z, X, DELTA, IPT, II, IM, M, ANG)

```
C      THIS SUBROUTINE FURNISHES THE TRANSFER FUNCTION
```

C TO BE REGRESSED ON BY R - D

```

DIMENSION X(1), Y(1), Z(1), ANG(1)

```

IF (IPT - M) 5,5,6

5 IF (IPT - II) 1,1,2

```
1 DELT11 = Y(IPT)*COS(ANG(IPT))
```

$$\text{DELTA12} = X(3) / ((1. - (Z(\text{IPT}) * X(1)) ** 2) ** 2 + (2. * X(2) * Z(\text{IPT})) ** 2) ** .5$$

```
DELT13 =COS(ATAN((-2.*Z(IPT)*X(2))/(1. -(Z(IPT)*X(1))**2)))
```

$$\text{DELT14} = \text{DELT13} * \text{DELT12}$$
$$\text{DELTA} = \text{DELT11} - \text{DELT14}$$

GO TO 9

```
2 DELT31 = Y(IPT)*COS(ANG(IPT))
```

$$\text{DELT32} = X(3) / ((1. - (Z(\text{IPT}) * X(1)) ** 2) ** 2 + (2. * X(2) * Z(\text{IPT})) ** 2) ** .5$$

```
DELT33 = ATAN((-2.*Z(IPT)*X(2))/(1.-(Z(IPT)*X(1))**2))
```

$$\text{DELT35} = \text{COS}(\text{DELT33} - 3.14159)$$
$$\text{DELT34} = \text{DELT35} * \text{DELT32}$$
$$\text{DELTA} = \text{DELT31} - \text{DELT34}$$

GO TO 9

6 IF (IPT - IM) 3,3,4

```
3 DELT21 = Y(IPT)*SIN(ANG(IPT))
```

```
DELTA22=X(3)/((1.-(Z(IPT)*X(1))**2)**2 + (2.*X(2)*Z(IPT))**2)**.5
```

```
DELTA23 = ATAN((-2.*Z(IPT)*X(2))/(1.-(Z(IPT)*X(1))**2))
```

$$\text{DELT24} = \text{SIN}(\text{DELT23})$$
$$\text{DELT25} = \text{DELT22} * \text{DELT24}$$

DELTA = DELT21 - DELT25

GO TO 9

```
4 DELT41 = Y(IPT)*SIN(ANG(IPT))
```

$$\text{DELT42} = X(3) / ((1. - (Z(\text{IPT}) * X(1)) ** 2) ** 2 + (2. * X(2) * Z(\text{IPT})) ** 2) ** .5$$

```
DELT43 = ATAN((-2.*Z(IPT)*X(2))/(1.-(Z(IPT)*X(1))**2))
```

$$\text{DELT44} = \text{SIN}(\text{DELT43} - 3.14159)$$
$$\text{DELT45} = \text{DELT42} * \text{DELT44}$$
$$\text{DELTA} = \text{DELT41} - \text{DELT45}$$

GO TO 9

CALC RESIDUE(S) FOR IPT DATA POINT
AND PRESENT X VALUES

NOTE..... ALSO KEEP NO. OF DATA PTS. ON LEFT OF WT =1.
EQUAL TO NO. OF DATA PTS. ON RIGHT OF WT =1.

NOTE..... ALWAYS KEEP M/2 TH PT. ON THE LEFT OF WT =1.

NOTEALWAYS START WITH X(1) BETWEEN
THE TWO MIDDLE POINTS

9 RETURN

END

```

SIBFTC 2124A  NODECK
      SUBROUTINE F2PCAL ( X, Q, ANL, Y, Z, N, M, P, II )
C      THIS SUBROUTINE FURNISHES ADDITIONAL INFORMATION
C      CONCERNING THE REGRESSED FUNCTION AND PARAMETERS
      DIMENSION X(6), Q(70), Z(70), ANL(70), P(70), Y(70)
      WRITE (6,190)
190  FORMAT (//20H1REGRESSION COMPLETE//)
      WRITE (6,191)
      X(2) = X(2)/X(1)
191  FORMAT (//16H CORRECT  X(I'S)//)
      WRITE (6,150) (X(I), I=1,N)
      WRITE (6,162)
162  FORMAT (//31H0Q CALCED AT EACH Z FOR NEW X'S//)
      DO 91 I=1,M
        91  Q(I)=X(3)/((1.-(Z(I)*X(1))**2)**2 + (2.*X(2)*Z(I)*X(1))**2)**.5
        WRITE (6,150) ( Q(I), I =1,M)
        WRITE (6,155)
155  FORMAT (//34H  CALCED PHASE ANGLES FOR NEW X(S)//)
      DO 68 I=1,M
        68  ANL(I) = ATAN((-2.*Z(I)*X(1)*X(2))/(1.-(Z(I)*X(1))**2))-3.14159
      DO 66 I=1,II
        66  ANL(I) = ATAN((-2.*Z(I)*X(1)*X(2))/(1. -(Z(I)*X(1))**2))
      DO 67 I=1,M
        67  ANL(I) = (ANL(I))*57.29578
        WRITE (6,150) (ANL(I), I=1,M)
        WRITE (6,163)
163  FORMAT (//22H0DIFFERENCE IN Q AND Y//)
      DO 11 I=1,M
        11  P(I) = Q(I) - Y(I)
        WRITE (6,150) ( P(I), I=1,M)
        WRITE (6,164)
164  FORMAT (//37H0DIFFERENCE IN Q AND Y ( NORMALIZED )//)
      DO 998 I=1,M
        998  Q(I) = ( Q(I) - Y(I) )/X(3)
        WRITE (6,150) ( Q(I), I=1,M )
        SQRQ = 0.0
        DO 12 I=1,M
          12  SQRQ = SQRQ + (Q(I))**2
          WRITE (6,152)
152  FORMAT (//45H0SUM OF SQUARES OF THE NORMALIZED DIFFERENCES//)
        WRITE (6,151) SQRQ
151  FORMAT (1X, E15.6)
150  FORMAT (1X, 10F13.8)
C
      RETURN
C
      END

```

APPENDIX III

FORTRAN PROGRAM LISTINGS

FOR SECTION IV

\$IBFTC 2014F

C PROGRAM TO NUMERICALLY REDUCE PULSE TEST INPUT AND OUTPUT DATA TO
C AMPLITUDE RATIO AND PHASE ANGLE AS FUNCTIONS OF FREQUENCY, USING
C PIECEWISE LINEAR APPROXIMATION AT HIGH FREQUENCIES AND TRAPEZOIDAL
C INTEGRATION AT LOW FREQUENCIES
C

DIMENSION FI(100), TI(100), FO(100), TO(100)
DIMENSION SI(100), SO(100)
DIMENSION TITLE(13)

C PI = 3.141592654
C ANGLK = 180. / PI

10 READ 1020, TITLE
PRINT 2060, TITLE

C READ 1000, NW, WMAX, NI, NO
C PRINT 2000, NW, WMAX, NI, NO

C READ 1010, (TI(J), FI(J), J = 1, NI)
C READ 1010, (TO(J), FO(J), J = 1, NO)

C PRINT 2010, (TI(J), FI(J), J = 1, NI)
C PRINT 2020, (TO(J), FO(J), J = 1, NO)

C NIM1 = NI - 1
C NOM1 = NO - 1
C DLOGW = (ALOG10(WMAX) + 1.) / FLOAT(NW - 1)
C FW = 10. ** DLOGW
C K = 0

C AI = 0.
C DO 12 J = 1, NIM1
12 AI = AI + (FI(J) + FI(J+1)) * (TI(J+1) - TI(J))

C AO = 0.
C DO 13 J = 1, NOM1
13 AO = AO + (FO(J) + FO(J+1)) * (TO(J+1) - TO(J))

C FNORM = AO / AI
C FNRMI = SQRT(FNORM)
C FNRMO = 1. / FNRMI

C DO 14 J = 1, NI
14 FI(J) = FI(J) * FNRMI

C DO 15 J = 1, NO
15 FO(J) = FO(J) * FNRMO

C PRINT 2030, FNRMI, FNRMO
C PRINT 2010, (TI(J), FI(J), J = 1, NI)
C PRINT 2020, (TO(J), FO(J), J = 1, NO)

C DO 16 J = 1, NIM1
16 SI(J) = (FI(J+1) - FI(J)) / (TI(J+1) - TI(J))

C DO 17 J = 1, NOM1
17 SO(J) = (FO(J+1) - FO(J)) / (TO(J+1) - TO(J))

```

C      PRINT 2045
C
C      18 PRINT 2040
C
C      W      = .1
C      DO 80 N = 1,NW
C
C      IF(K .EQ. 1) GO TO 40
C
C      AI      = 0.
C      BI      = 0.
C      WT      = W * TI(1)
C      FCOSJ   = FI(1) * COS(WT)
C      FSINJ   = FI(1) * SIN(WT)
C      DO 20 J = 1,NIM1
C      WT      = W * TI(J+1)
C      FCOSJ1  = FI(J+1) * COS(WT)
C      FSINJ1  = FI(J+1) * SIN(WT)
C      DT      = TI(J+1) - TI(J)
C      AI      = AI + (FCOSJ + FCOSJ1) * DT
C      BI      = BI + (FSINJ + FSINJ1) * DT
C      FCOSJ   = FCOSJ1
20    FSINJ   = FSINJ1
C      AI      = .5 * AI
C      BI      = .5 * BI
C
C      AO      = 0.
C      BO      = 0.
C      WT      = W * TO(1)
C      FCOSJ   = FO(1) * COS(WT)
C      FSINJ   = FO(1) * SIN(WT)
C      DO 30 J = 1,NOM1
C      WT      = W * TO(J+1)
C      FCOSJ1  = FO(J+1) * COS(WT)
C      FSINJ1  = FO(J+1) * SIN(WT)
C      DT      = TO(J+1) - TO(J)
C      AO      = AO + (FCOSJ + FCOSJ1) * DT
C      BO      = BO + (FSINJ + FSINJ1) * DT
C      FCOSJ   = FCOSJ1
30    FSINJ   = FSINJ1
C      AO      = .5 * AO
C      BO      = .5 * BO
C
C      GO TO 70
C
C      40 WI      = 1. / W
C
C
C      AI1      = 0.
C      AI2      = 0.
C      BI1      = 0.
C      BI2      = 0.
C      WT      = W * TI(1)
C      COSJ     = COS(WT)
C      SINJ     = SIN(WT)
C      FCOSJ    = FI(1) * COSJ
C      FSINJ    = FI(1) * SINJ
C      DO 50 J = 1,NIM1

```

```

      WT      = W * TI(J+1)
      COSJ1   = COS(WT)
      SINJ1   = SIN(WT)
      FCOSJ1  = FI(J+1) * COSJ1
      FSINJ1  = FI(J+1) * SINJ1
      AI1     = AI1 + FSINJ1 - FSINJ
      AI2     = AI2 + SI(J) * (COSJ1 - COSJ)
      BI1     = BI1 + FCOSJ1 - FCOSJ
      BI2     = BI2 + SI(J) * (SINJ1 - SINJ)
      COSJ    = COSJ1
      SINJ    = SINJ1
      FCOSJ   = FCOSJ1
50  FSINJ    = FSINJ1
      AI      = WI * (AI1 + WI * AI2)
      BI      = -WI * (BI1 - WI * BI2)

C
      AO1     = 0.
      AO2     = 0.
      BO1     = 0.
      BO2     = 0.
      WT      = W * TO(1)
      COSJ    = COS(WT)
      SINJ    = SIN(WT)
      FCOSJ   = FO(1) * COSJ
      FSINJ   = FO(1) * SINJ
      DO 60 J = 1,NOM1
      WT      = W * TO(J+1)
      COSJ1   = COS(WT)
      SINJ1   = SIN(WT)
      FCOSJ1  = FO(J+1) * COSJ1
      FSINJ1  = FO(J+1) * SINJ1
      AO1     = AO1 + FSINJ1 - FSINJ
      AO2     = AO2 + SO(J) * (COSJ1 - COSJ)
      BO1     = BO1 + FCOSJ1 - FCOSJ
      BO2     = BO2 + SO(J) * (SINJ1 - SINJ)
      COSJ    = COSJ1
      SINJ    = SINJ1
      FCOSJ   = FCOSJ1
60  FSINJ    = FSINJ1
      AO      = WI * (AO1 + WI * AO2)
      BO      = -WI * (BO1 - WI * BO2)

C
70  AMPL     = SQRT((AO**2 + BO**2) / (AI**2 + BI**2))
      ARG    = (AO*BI - AI*BO) / (AO*AI + BO*BI)
      ANGLR   = ATAN(ARG)
      IF(ARG .GT. 0.) ANGLR = ANGLR - PI
      ANGLD   = ANGLR * ANGLK
      WLOG    = ALOG10(W)

C
      PRINT 2050, W, WLOG, AMPL, ANGLR, ANGLD, AO, BO, AI, BI

C
80  W        = FW * W

C
      K      = K + 1
      IF(K .EQ. 2) GO TO 10
      PRINT 2046
      GO TO 18

C

```

```

C
1000 FORMAT(I5, 5X, F10.0, I5, 5X, I5)
C
1010 FORMAT(8F10.0)
C
1020 FORMAT(13A6, A2)
C
2000 FORMAT(1H0, 18HNO. OF FREQUENCIES, I11 / 18H0MAXIMUM FREQUENCY,
1 F12.5 / 20H0NO. OF INPUT POINTS, I10 / 21H0NO. OF OUTPUT POINTS,
2 I9 )
C
2010 FORMAT(1H0 /// 18H0INPUT DATA POINTS / 1H0, 8X, 4HTIME, 12X,
1 11HPULSE LEVEL // (1H , 2F20.6))
C
2020 FORMAT(1H0 /// 19H0OUTPUT DATA POINTS / 1H0, 8X, 4HTIME, 12X,
1 11HPULSE LEVEL // (1H , 2F20.6))
C
2030 FORMAT(1H0 /// 25H0INPUT NORMALIZING FACTOR, F10.5/
1 26H0OUTPUT NORMALIZING FACTOR, F9.5 /// 16H0NORMALIZED DATA)
C
2040 FORMAT(1H0 /// 23H0DATA REDUCTION RESULTS //
1 1H0, 6X, 5H0OMEGA, 8X, 9HLOG OMEGA, 6X, 9HAMPLITUDE, 6X,
1 9HANGLE-RAD, 6X, 9HANGLE-DEG, 9X, 2HA0, 12X, 2HBO, 12X, 2HAI,
1 12X, 2HBI / )
C
2045 FORMAT(1H1, 23HTRAPEZOIDAL INTEGRATION )
2046 FORMAT(1H1, 30HPIECEWISE LINEAR APPROXIMATION )
C
2050 FORMAT(1H , 5F15.6, 4F14.6)
C
2060 FORMAT(1H1, 13A6, A2)
C
END

```